Universidad Autónoma de Madrid

Escuela Politécnica Superior

Grado en Ingeniería Informática

TRABAJO DE FIN DE GRADO

APLICACIÓN MÓVIL Y WEB PARA EL CONTROL DE ASISTENCIA DE LOS EMPLEADOS DE UNA COMPAÑÍA

Carlos Martí González
Tutor: José Antonio Clavijo Blázquez
Ponente: Simone Santini

Noviembre 2019

APLICACIÓN MÓVIL Y WEB PARA EL CONTROL DE ASISTENCIA DE LOS EMPLEADOS DE UNA COMPAÑÍA

AUTOR: Carlos Martí González
TUTOR: José Antonio Clavijo Blázquez
PONENTE: Simone Santini

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid

Noviembre 2019

Resumen

Resumen — Actualmente, la expansión de las aplicaciones móviles y web tiene una progresión exponencial. Por este motivo, muchas de las plataformas tienden a tener una aplicación web accesible desde cualquier navegador, así como una aplicación móvil, aumentando el impacto del negocio y obteniendo una mayor expansión.

Este Trabajo de Fin de Grado ha constado en realizar dos aplicaciones para el control de horas, una para móvil y, otra para web. Se ha desarrollado en la empresa Delonia Software SL. la cual ha propuesto este proyecto para una gestión interna del registro de horas laborales de los empleados como una alternativa a la actual manera de llevar este control. Tendrá disponibilidad tanto en Android como en iOS, así como acceso desde cualquier navegador independientemente del sistema operativo. Las aplicaciones van dirigidas, por una parte, a los empleados de Delonia que podrán gestionar el registro de sus horas y, por otro lado, a Delonia como empresa, para poder llevar un control de las horas conforme a la nueva legislación con la posibilidad de gestionarlas.

La motivación de esta propuesta viene a raíz de un análisis previo realizado sobre la forma en la que se realizaba el control de horas. Durante los meses que lleva implantado este sistema, se han hecho evidentes algunas de los defectos y carencias que acarrea el uso de una hoja de cálculo para este objetivo. Se ha podido ver que dicha solución carece de una seguridad mínima que garantice la veracidad de los registros, además de falta de automatización del proceso de firmar y modificar datos, debido a que existe la necesidad de tener que imprimir los registros mensualmente.

El proyecto, llamado Tracker, permite el control de horas por parte de los empleados, iniciando sesión con el correo de empresa asociado. Además, permite la modificación de los registros incluyendo la eliminación de los mismos. Todo ello, tanto desde la aplicación móvil (iOS y Android) como en la aplicación web accediendo desde un navegador, consiguiendo un alcance casi absoluto para los empleados.

Este proyecto ha ido pasando por una serie de etapas secuencialmente, todas ellas imprescindibles para conseguir el producto final deseado. Se ha realizado un análisis completo para definir y comprobar la viabilidad del proyecto. Posteriormente, se ha llevado un diseño de la arquitectura, de la interfaz gráfica y de la base de datos, basándose en los requerimientos definidos en la etapa anterior. Finalmente, la última etapa ha consistido en el desarrollo de las dos aplicaciones y el servidor, incluyendo la codificación y las pruebas realizadas.

Por último, todo el proyecto ha sido supervisado por un ingeniero y por un arquitecto de software. El proyecto se compone del *back-end* (base de datos y servidor), como *front-end* (aplicación web y móvil). Utilizando para las aplicaciones web y móvil, `React` en JavaScript y `Xamarin Forms` en C#, respectivamente. Para la parte del *back-end* se ha utilizado `PostgreSQL` como gestor de base de datos, `Tomcat` como contenedor web y `Uaithne` para el desarrollo del servidor.

Palabras clave — Web, Xamarin Forms, React, JavaScript, C#, App móvil, iOS, Android, Navegador, horas, empleado, administrador, OpenSource, BD, NFC, control, HTML, CSS, ley, Java, Uaithne, back-end, front-end

Abstract

Abstract — Currently, the expansion of mobile applications and web applications has an exponential progression. For this reason, many of the platforms tend to have a web application accessible from any browser, as well as a mobile application increasing the impact of the business in addition to its greater expansion.

This end-of-degree Project has included two applications for the control of hours, one for mobile and another for web. It has been developed in the company Delonia Software SL. proposing this project for an internal management of the employee's working hours register as an alternative to the current way of keeping this control. You will have availability on both Android and iOS, as well as access from any browser. The applications are directed, on the one hand, to Delonia employees who may register their hours and, on the other hand, to Delonia as a company, in order to keep track of the hours in accordance with the new legislation with the possibility of managing them.

The motivation of this proposal comes as a result of a previous analysis carried out on the current form of the hours register. During the months that this system has been implemented, some of the defects and deficiencies caused by the use of a spreadsheet for the control of working hours have become evident. It has been seen that said solution lacks a minimum security that guarantees the veracity of the records, in addition to the lack of automation of the process of signing and modifying data, because there is a need to have to print the data monthly.

The project, called Tracker, allows the registration of hours by employees, by logging in with the associated company mail. In addition, it allows the modification of the records including the elimination of these. All this, both from the mobile application (iOS and Android) and in the web application accessing from a browser, getting an almost absolute reach for employees.

This project has been going through a series of stages sequentially, but all essential for the achievement of the desired final product. An exhaustive analysis has been carried out to define and verify the viability of the project. Subsequently, an architecture, interface and database design has been carried out, based on the requirements defined in the previous stage. Finally, the last stage has been the development of applications, which include coding and testing.

Finally, the entire project has been supervised by a computer engineer and a software architect. The project consists of the *back-end* (database and server), such as *front-end* (web and mobile application). Using for web and mobile applications, `React` in JavaScript and `Xamarin Forms` in C #, respectively. For the part of the *back-end* `PostgreSQL` has been used as a database manager, `Tomcat` as a web container and `Uaithne` for server development.

Key words — Web, Xamarin Forms, React, JavaScript, C #, Mobile App, iOS, Android, Browser, hours, employee, administrator, OpenSource, BD, NFC, control, HTML, CSS, law, Java, Uaithne,back-end, front-end

Agradecimientos

A mis abuelas, ayudándome siempre que lo he necesitado, y dándome esos consejos para la vida que solo ellas saben y guiándome siempre.

A mis padres, por permitirme el acceso a la educación que me han dado, por enseñarme los valores de que con esfuerzo y, la perseverancia se consigue lo que te propongas, y, sobre todo, por enseñarme a no parar de levantarme tras cada caída.

A mi hermana y mi hermano, que me han podido ayudar en los momentos malos que he tenido en mi vida, apoyándonos siempre que podemos.

A mi pareja Sara, que este último año me ha ayudado a poder seguir con la misma fuerza que siempre, por el apoyo y su inmensa paciencia y, que en este trabajo me ha animado siempre para dar lo mejor de mi.

A mis amigos del colegio, sabiendo que durante esta etapa me ha sido complicado poder compaginar estudios y trabajo y, aún así, siempre estaban con una sonrisa para ver que tal iba todo.

A mis amigos de la universidad, que en estos cuatro años hemos sufrido juntos todas las prácticas y las asignaturas difíciles, pero también disfrutando de la vida de universitarios como se merece.

A los profesores del Grado de Ingeniería Informática de la EPS, en la Universidad Autónoma de Madrid, por haberme formado y haberme dado la base con la que poder desenvolverme en el ámbito laboral y por enseñarme a aprender.

A Delonia Software SL. por haberme podido dar la oportunidad de trabajar con ellos, haber confiado en mi y en mis capacidades, además de poder formar parte de un gran equipo.

"Sic Parvis Magna"
Sir Francis Drake

Índice general

Índice de tablas

Índice de figuras

1 Introducción

En este primer capítulo se explica el tema abordado en el proyecto dando así una idea global. Se explica la motivación que ha llevado a cabo este tema, los objetivos que debe cumplir el proyecto y la estructura de la que se compone.

1.1 Motivación

Esta memoria de Trabajo de Fin de Grado (TFG) titulada ‘‘`Aplicación móvil y web para el control de asistencia de los empleados de una compañía`’’ surge a raíz de la nueva normativa aprobada en el artículo 34 del Estatuto de los Trabajadores, en el cual se obliga a regular y controlar el registro de las horas de trabajo de los empleados de una empresa. Para ello, tienen que quedar reflejadas las horas de entrada y salida, el tiempo invertido en comidas y descanso, y las horas totales que se han hecho cada día.

Para cumplir esta normativa, muchas empresas han tenido que adaptarse e implantar sistemas que, en muchos casos, son rudimentarios y poco eficientes. Algunos de estos, van desde hojas de Excel hasta hojas en papel. Son soluciones fáciles, pero poco eficientes y sin ninguna certificación externa de que se cumplen dichas horas.

Con todo esto, surge la necesidad de crear un sistema económicamente viable, pero lo suficientemente seguro para poder certificar la veracidad de las horas registradas. Por tanto, la solución propuesta consiste en una aplicación móvil y dos aplicaciones web (una, para empleados; otra, para administradores) con la posibilidad de utilizar algún tipo dispositivo `hardware`. Es una solución alternativa a las actuales consiguiendo un sistema mejor y más tecnológico.

Esta solución implica que las motivaciones vayan más allá de cumplir la nueva normativa. Hay tres motivaciones principales en esta propuesta: motivación tecnológica, motivación de negocio y motivación organizativa.

1.1.1. Motivación de negocio

Una de las motivaciones que existen es la posibilidad de obtener una reducción de costes en lo que se refiere a la gestión que se lleva a cabo para el registro de horas. Esta reducción viene a través de la automatización y simplificación de los procesos de registro y almacenamiento de los datos. Además, también existe una reducción del tiempo empleado en su gestión debido a la mejora del sistema y su facilidad de uso.

Otro de los aspectos que tiene como objetivo este proyecto, es tener la posibilidad de una expansión en el mercado, es decir, poder ofertar las aplicaciones a otras empresas con la misma problemática. Esta oferta puede llegar desde vender las aplicaciones, hasta la gestión y administración completas para otras empresas.

1.1.2. Motivación organizativa

Otra de las grandes motivaciones de esta propuesta es debido a que el modelo que hay actualmente es tedioso y poco intuitivo, además de bastante rudimentario. Por ese motivo, la solución propuesta pretende cumplir de una manera más clara y concisa las normativas y estándares actuales, para mayor facilidad, comodidad y claridad.

Además, hay que recalcar que en lo referente al acceso de los datos se mejora de una manera exponencial ya que se almacenan en una base de datos (BD), normalizando su acceso. Esto conlleva que la consulta de la información sea más sencilla permitiendo así una variada generación de informes.

Por otro lado, se gana en agilidad y automatización debido a la posibilidad de uso de dispositivos externos que permiten la verificación del registro de horas, ayudando a la gestión y control de los datos almacenados.

1.1.3. Motivación tecnológica

Por último, existe una motivación tecnológica debido a que los procesos de algunas empresas son bastante antiguos. Con esta propuesta se quiere tener acceso a una tecnología que pueda hacer mejorar los sistemas que se presentan actualmente haciendo uso de las grandes posibilidades que nos da la tecnología y su evolución constante.

Con este fin, existe una gran motivación por hacer uso de lo último en tecnología para el desarrollo de aplicaciones, pudiendo aprender e investigar sobre lo nuevo, así como tener la posibilidad de mejorar el desarrollo tanto de los actuales como de los futuros proyectos de la empresa.

1.2 Objetivos y enfoque

El objetivo que se quiere alcanzar con este proyecto es la evolución y mejora del sistema para el control de horas laborales de los empleados de una empresa. Para ello, se van a implementar una aplicación móvil y dos aplicaciones web, ayudadas opcionalmente por algún dispositivo `hardware` (tarjetas `NFC`, lectores de huella digital, `beacons`...) para poder certificar la localización del

empleado. Con el proyecto ya realizado y, progresivamente, se irá implantando en la empresa con el fin de poder reemplazar el sistema actual de manera total.

Los objetivos principales se pueden desglosar en los siguientes puntos:

- **Modernizar.** Renovar el modo de fichaje empleando aplicaciones móviles, permitiendo que el sistema funcione también ante situaciones en las que el trabajo se desarrolle fuera de la oficina.
- **Automatizar.** Mecanizar el sistema en conjunto con el posible uso de sistemas biométricos tales como lectores de huellas digitales; así como también otros sistemas basados en NFC o geolocalización pudiendo, de esta manera, obtener los datos de manera más sencilla y actualizada.
- **Mejorar.** Al cambiar el modo de almacenamiento en base de datos se mejora la calidad de los datos debido a que se pueden generar todo tipo de informes con la información que, de manera análoga, es inviable realizarse en el modelo actual. Asimismo, con este nuevo proceso, tanto la calidad como la claridad pueden llegar a mejorar sustancialmente.

Por tanto, la propuesta pretende dar un mejor enfoque para el proceso de registro en la empresa, unificándolo y clarificándolo para ayudar al cumplimiento de la normativa actual.

1.3 Metodología y Plan de Trabajo

La metodología de desarrollo de software se puede definir como un conjunto de técnicas, recomendaciones y verificaciones, que permitan sistematizar los procesos en los que se descompone el proyecto [1].

Antes de comenzar, por tanto, se tiene que poder establecer un plan de trabajo que se va a utilizar durante el tiempo que dure el proyecto. Para este proyecto se ha decidido usar una metodología ágil llamada *Scrum* siendo muy usada en proyectos de este estilo.

Scrum se rige por un proceso iterativo que reparte el desarrollo de un producto en distintas iteraciones de unas pocas semanas llamadas *sprints*. En cada *sprint* el equipo trabaja sobre una lista de requisitos previamente acordados obteniendo así, al acabar la iteración, un producto entregable. Este enfoque permite optimizar el control de riesgos y aumentar la previsibilidad debido a que, al tener un producto que pueda ver el cliente, se puede proporcionar un mejor *feed back* y comprobar si se están cumplimento sus expectativas [2].

Los equipos de desarrollo de *Scrum* son autogestionados y multifuncionales, es decir, en cada equipo debe de haber alguno de los miembros que sepa realizar cada una de las tareas a hacer. Los roles básicos de Scrum son los cuatro que se muestran en la Figura 1.1.

El rol de *Product Owner* es la persona que representa al *Cliente*. Es el responsable de organizar las diferentes tareas que se deben realizar, priorizando las más importantes y gestionando las distintas características del producto. Este rol lo ha desempeñado el director de la empresa.

El rol de *Scrum Master* es la persona que se encarga de la gestión del proyecto. Es el responsable de que el *Scrum Team* entienda lo que tienen que hacer además de asegurarse de que haya un buen ambiente de trabajo. Este rol lo ha desempeñado un empleado de Delonia Software designado a este proyecto.

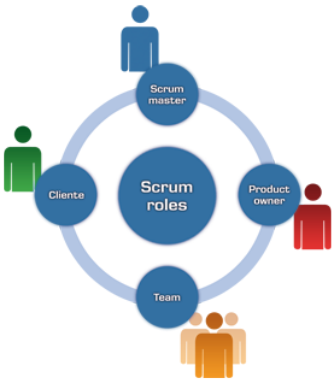

Figura 1.1: Roles de Scrum.
Fuente: softeng.es

Por último, el rol de *Scrum Team* es el equipo encargado de desarrollar el proyecto. No hay una jerarquía interna debido a que el equipo se debe autogestionar. Este es el rol que yo he desempeñado en el proyecto.

En la Figura 1.2, se puede apreciar el flujo principal que sigue tipo de metodología.

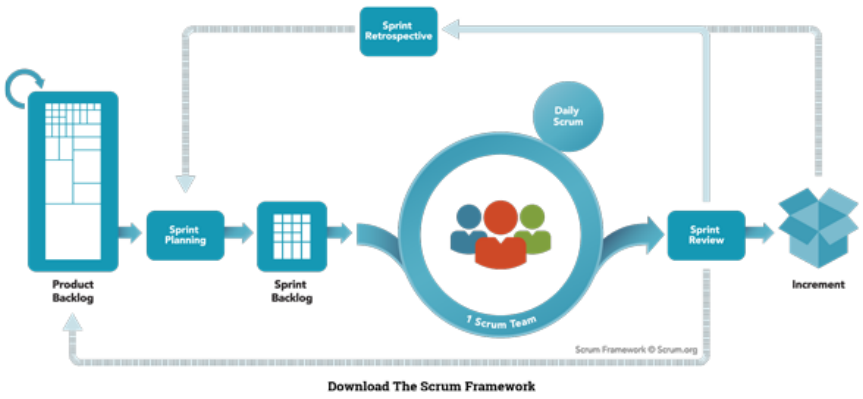

Figura 1.2: Metodología Scrum.
Fuente: scrum.org

Por tanto, el proyecto ha sido abarcado por un total de tres personas que, siendo un equipo tan reducido la metodología de *Scrum* que se ha aplicado ha sido un poco más simplificada, pero ha seguido el esquema general que se puede ver en la Figura 1.2. Aún siendo algo más simple, también se han realizado *daily meeting's* donde se iba transmitiendo el estado de las tareas asignadas, los posibles bloqueos y las tareas ya finalizadas.

También, se ha realizado un *Sprint Backlog* (Figura 1.3), donde se exponían las tareas que se estaban realizando de manera gráfica, y se informaba de las horas que se iban a tardar en realizar, actualizándola en cada *daily*.

Finalmente y, en conjunto con el *Sprint Backlog* se ha ido actualizando la llamada *Burndown Chart*. Es una gráfica que muestra la evolución diaria en el número de horas restantes para finalizar las tareas asociadas a ese *sprint*. Consta de dos líneas, una es una recta que representa la evolución teórica, la otra es la evolución real de las horas utilizadas diariamente. En la Figura

1.4 se muestra un el gráfico de quemado de una semana del proyecto.

Es verdad, que la metodología *Scrum* está orientada a trabajar en equipos de desarrollo algo más numerosos y, no solo una persona como es el caso, pero se han aprovechado una serie de características que tiene esta metodología para aplicarlas en este proyecto. Como se ha explicado anteriormente, tanto las *daily's* como la utilización de un *Backlog* han servido mucho a la hora de la organización.

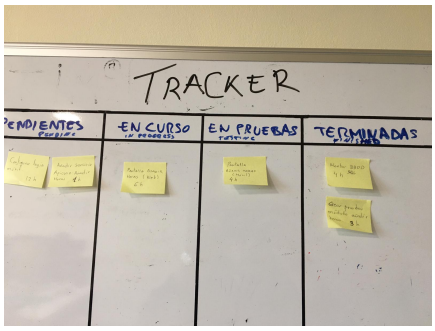

Figura 1.3: Backlog de un día.

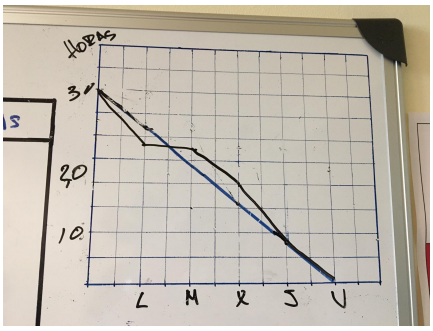

Figura 1.4: Burndown Chart de un sprint.

Fuente: Propia

En referencia al control de versiones que se ha llevado en el proyecto, se ha usado Git para la gestión del software como es la gestión del código, de los documentos y el resto de archivos que conforman el proyecto. Como pasa con *Scrum* su uso esta pensado para un grupo de personas más o menos grande, que compartan los archivos del proyecto, para que, de una manera más eficiente y fiable, se tenga un control en todos los archivos del proyecto, evitando conflictos. El uso de Git se ha llevado a cabo a partir de una herramienta llamada SourceTree.

SourceTree [3] es una herramienta que permite llevar una gestión de cualquier repositorio Git, a través de un programa muy simple e intuitivo. Su principal característica es la simplicidad con la que se muestran los comentarios y el historial de cambios, debido a que son líneas con una proyección muy concreta. Además, es muy intuitivo el modo en que se puede subir, bajar archivos, eliminar comentarios y poder ver que es lo que has cambiado, debido a que automáticamente puedes ver los cambios realizados de manera local. Como se puede ver en la Figura 1.5, es una interfaz muy sencilla e intuitiva.

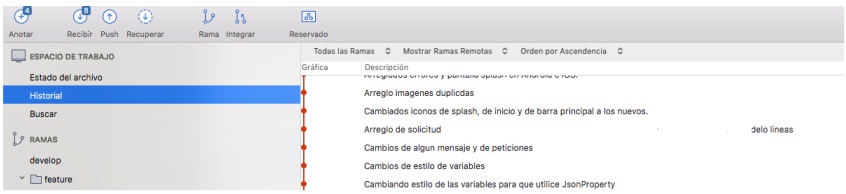

Figura 1.5: Aplicación de SourceTree.

Fuente: Propia

1.4 Estructura del documento

El trabajo presentado se ha estructurado siguiendo las distintas etapas básicas que sigue un proyecto de desarrollo de software.

- En el *Capítulo 1* se explica una introducción al proyecto planteado, así como diferentes aspectos generales incluyendo la metodología y la estructura del trabajo.
- En el *Capítulo 2* se habla sobre la nueva ley que regula el registro de las horas laborales, describiendo el ámbito, los motivos y la forma en que las empresas deben gestionarlo. Además, se habla de la actualidad sobre las aplicaciones móviles y las páginas web, enumerando algunas aplicaciones similares que existen en la actualidad.
- En el *Capítulo 3* se muestra y detalla el análisis realizado, incluyendo tanto el análisis funcional como el alcance esperado para proyecto.
- En el *Capítulo 4* se expone la etapa de diseño en la que se recoge el patrón utilizado para el diseño, la arquitectura así como la base de datos, maquetación y las herramientas empleadas para su diseño.
- En el *Capítulo 5* se presenta la etapa de desarrollo del proyecto en la que detallan las tecnologías y herramientas utilizadas para la elaboración del proyecto.
- En el *Capítulo 6* se recogen las pruebas realizadas durante todo el proyecto, indicando los diferentes tipos.
- En el *Capítulo 7* se disponen las conclusiones extraídas del proyecto además de las perspectivas para el trabajo futuro.

2

Estado del Arte

2.1 Introducción

En este capitulo se detallará algunos de los aspectos actuales que tienen relación con el proyecto. En concreto, se hablará sobre la legislación que regula el control del horario laboral y los motivos por los que se obliga a hacerlo, la actualidad sobre las aplicaciones móviles y las páginas Web, y, por último se expondrán las aplicaciones similares que hay actualmente en el mercado.

Por tanto, se explicará el contexto y las tecnologías que existen del proyecto que estamos trabajando y así poder justificar las decisiones tomadas a lo largo de este.

2.2 Real Decreto-ley 8/2019

El Real Decreto-ley [4] tiene como objetivo principal poder potenciar las políticas sociales incluyendo determinadas disposiciones que están dirigidas a establecer el registro de la jornada de trabajo. Estas medidas garantizan que se cumplen los límites en lo que a la jornada respecta, creando un marco de seguridad jurídica tanto para los trabajadores como para las empresas, habilitando la posibilidad del un control por parte de la Inspección de Trabajo y Seguridad Social.

Las reglas sobre la limitación de las horas de la jornada laboral se configuran como un elemento de protección de los trabajadores y se agrupan en torno al cumplimiento de estas. La realización de un tiempo de trabajo mayor de la jornada laboral legal afecta en dos aspectos esenciales para la relación laboral como son la vida personal y el salario.

Una de las causas que han incidido en los problemas del control de la jornada, ha sido la ausencia en el Estatuto de los Trabajadores de la obligación del registro de la jornada por parte de la empresa. Esto ha supuesto dificultades a la hora de interponer reclamaciones por parte de

los trabajadores a los que se les exige la extralimitación de sus horas correspondientes.

Actualmente, existen normas que permiten una flexibilidad horaria permitiendo adaptarse a las necesidades particulares de la empresa, del mercado o del trabajador. Esta flexibilidad permitía, con la justificación de ser variable, un exceso o defecto de horas realizadas y, por tanto, una serie de conflictos entre la empresa y el trabajador debido a que uno de los dos se ve afectado por alguno de estos casos.

A la hora de justificar las medidas referidas al registro de la jornada, debe contarse que, durante el 2018, un 35 % de las denuncias en materias laborales se referían al tiempo de trabajo. Además, utilizando algunos datos de la Encuesta de Población Activa, cada semana del año 2017, se hicieron, en España, 5,8 millones de horas extra de media. Esto, añadido al 48 % de los trabajadores que declaran realizar horas extra que no les son pagadas, suponiendo un perjuicio grave para las personas afectadas.

Por ello, a través de la modificación del articulo 34 que regula el registro de jornada para poder garantizar el límite de horas y facilitar la resolución de las discrepancias generadas en cuanto al tiempo de las jornadas. También, se modifica el artículo 7 para poder tipificar como infracciones las derivadas del incumplimiento del registro. Las modificaciones en los diferentes artículos son las expuestas a continuación:

- *"34.7. El Gobierno, a propuesta de la persona titular del Ministerio de Trabajo, Migraciones y Seguridad Social y previa consulta a las organizaciones sindicales y empresariales más representativas, podrá establecer ampliaciones o limitaciones en la ordenación y duración de la jornada de trabajo y de los descansos, así como especialidades en las obligaciones de registro de jornada, para aquellos sectores, trabajos y categorías profesionales que por sus peculiaridades así lo requieran"*

- *"34.9. La empresa garantizará el registro diario de jornada, que deberá incluir el horario concreto de inicio y finalización de la jornada de trabajo de cada persona trabajadora, sin perjuicio de la flexibilidad horaria que se establece en este artículo.*
  Mediante negociación colectiva o acuerdo de empresa o, en su defecto, decisión del empresario previa consulta con los representantes legales de los trabajadores en la empresa, se organizará y documentará este registro de jornada.
  La empresa conservará los registros a que se refiere este precepto durante cuatro años y permanecerán a disposición de las personas trabajadoras, de sus representantes legales y de la Inspección de Trabajo y Seguridad Social"

- *"7.5. La transgresión de las normas y los límites legales o pactados en materia de jornada, trabajo nocturno, horas extraordinarias, horas complementarias, descansos, vacaciones, permisos, registro de jornada y, en general, el tiempo de trabajo a que se refieren los artículos 12, 23 y 34 a 38 del Estatuto de los Trabajadores"*

En lo que se refiere a cómo debe de realizarse el registro, la normativa no especifica un mecanismo en concreto. El único contexto es que se debe realizar mediante una negociación colectiva. Además, es responsabilidad de la empresa el garantizar el registro diario así como cumplir también los requisitos contemplados en la ley de protección de datos, cerciorándose de que el mecanismo usado para tal fin lo cumple [5].

2.3 Aplicaciones Web

Las aplicaciones web son programas que los usuarios pueden tener acceso a través de una red local o a través de Internet mediante cualquier navegador web, es decir, son herramientas que se codifican en un lenguaje que interpretan y ejecutan los navegadores. A través de ellos, se puede acceder a la funcionalidad y se puede tener todo tipo de servicios como puede ser correos, wikis, blogs, tiendas o simples páginas web informativas [6].

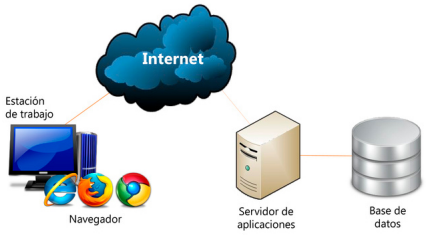

Figura 2.1: Esquema aplicación web.
Fuente: neosoft.es

Como se puede ver en la Figura 2.1, el funcionamiento de una aplicación web es sencillo. Los datos y la información, se almacenan en bases de datos, formadas por un número de tablas. La aplicación solicita una información a la base de datos que, a través del servidor, es transmitida y devuelta a la aplicación web.

Las aplicaciones web deben su popularidad a los navegadores web debido a que son clientes ligeros. Tienen total independencia del sistema operativo, además de ser simples para su actualización, mantenimiento y gestión. Cabe mencionar que una aplicación web, puede tener elementos que permitan una comunicación activa entre el usuario y el sistema de información, de manera que la página responderá a cada una de las acciones realizadas por el usuario.

En los inicios de la arquitectura cliente-servidor, cada usuario tenía instalado en su computadora una aplicación cliente que se conectaba al servidor. Esto producía que, si había una mejora en el servidor, tendría que haber una actualización en todos los programas de los clientes instalados.

En contraste, las aplicaciones web generan de manera dinámica las distintas páginas en un formato que es soportado por los navegadores como puede ser HTML o XHTML. Además, se utilizan lenguajes interpretados, no compilados, para añadir elementos dinámicos a través de lenguajes como `JavaScript`, `Java`, etc. En referencia, al estilo que va a seguir la página, se utiliza un lenguaje de diseño gráfico conocido como CSS, que sirve para definir y crear la presentación de un documento [7].

Como todos los programas y soluciones, las aplicaciones web conllevan una serie de ventajas y desventajas. A continuación, se detallan algunas de ellas:

- **Ventajas:**
  - Ahorro de tiempo ya que es posible realizar tareas simples sin instalar ningún programa externo.
  - No ocupa espacio en disco duro.
  - Se pueden usar en cualquier sistema operativo del cliente.

  - Posibilidad de las actualizaciones inmediatas.
  - Basta con tener un navegador web.

- **Inconvenientes:**

  - Ofrecen menos rendimiento que las propias aplicaciones de escritorio al tratarse de sistemas interpretados.
  - La disponibilidad de la aplicación depende de un proveedor de Internet o el que provee el servidor.

2.3.1. Multi-Page Application

Las *Multi-Page Application* (MPA) [8] son las que funcionan de forma "tradicional", es decir, cada cambio en la interfaz como, por ejemplo, mostrar los datos o enviarlos de vuelta, se representan como una nueva página del servidor en el propio navegador.

Estas aplicaciones, por lo general, son más grandes que las *Single-Page* debido a que, por la gran cantidad de contenido que manejan, se deben crear muchos niveles de interfaz. Anteriormente, esto era un problema ya que era necesario transferir muchos datos entre el servidor y el navegador.

En la actualidad, se han introducido diferentes tecnologías que hacen que no tengamos que preocuparnos de cargar la página entera cada vez que cambian los datos. En vez de ello, estas nuevas tecnologías, como `AJAX`, nos permiten actualizar exclusivamente partes de la aplicación. De esta manera, se mejora la solución y evita tener que transferir gran cantidad de datos entre el servidor y el navegador.

Por el contrario, el desarrollo de la aplicación se vuelve bastante complejo debido a que es necesario el uso de marcos tanto para el cliente, como para el servidor. Además, otro de los inconvenientes es debido a que la implementación entre el *front-end* y el *back-end* está muy acoplada.

Algunos ejemplos de *frameworks* que incluyen MPA son los siguientes [9]

- Java Spring
- ASP.NET MVC/Razor
- Python Django
- PHP Laravel

2.3.2. Single-Page Application

Las *Single-Page Application* (SPA) [10] son las aplicaciones web que tienen como objetivo poder mostrar una experiencia más fluida a los usuarios, asemejándose más a una aplicación de escritorio que a un navegador web, cargando únicamente una sola página.

Esto se debe a que, todos los códigos HTML, JavaScript y CSS solo se cargan una vez, empleando dinámicamente los recursos necesarios en respuesta a las acciones del usuario, es decir, solo se actualizan los fragmentos de las nuevas páginas. De esta manera, existe una analogía

entre los estados de un SPA y las páginas de un MPA, por lo que cualquier página MPA podría convertirse en un SPA, reemplazando únicamente las partes donde se genera el cambio [11].

En este tipo de aplicaciones, el desarrollo se simplifica ya que no hay que escribir código para la representación del servidor. Además, son fáciles de depurar y pueden almacenar en caché cualquier almacenamiento local que se necesite de manera efectiva.

En contraposición a estas ventajas, la descarga de datos es lenta debido a que tiene que cargar todo el marco del cliente en el navegador. Conjuntamente a esto, requiere tener habilitado JavaScript ya que es necesario para el intercambio de datos y actualización de la aplicación.

Finalmente, en lo que a seguridad se refiere y en comparación con las MPA, estas aplicaciones son menos seguras. El motivo de esto se basa en *Cross-Site Scripting* (XSS) que permite a los atacantes inyectar *scripts* del lado del cliente en la aplicación web.

Todos los *frameworks* que desarrollan estas aplicaciones están implementados en `JavaScript`. Algunos de ellos se muestran a continuación [9]:

- Angular JS
- React JS
- Vue JS
- Ember JS
- Meteor JS

Por estos motivos, se ha decantado por usar un entorno de tipo SPA, escogiendo en este caso `React JS` para el desarrollo.

2.4 Aplicaciones Móviles

En la actualidad, el número de aplicaciones móviles que existen cada vez es mayor, solo hay que tener presente el número de aplicaciones en plataformas como *App Store* y *Play Store*, donde hay un total de 1,8 y 2,4 millones de aplicaciones respectivamente [12].

Pero no existe una única manera de desarrollarlas, por lo que se pueden clasificar dependiendo de la manera en la que se haya desarrollado y para que plataformas [13].

2.4.1. Aplicaciones móviles Nativas

Las aplicaciones nativas [14] son aquellas desarrolladas exclusivamente para un solo sistema operativo (iOS, Android o Windows Phone). Esto implica el uso de un lenguaje específico que es compatible únicamente con cada uno de ellos. Por ejemplo, para iOS se utiliza *Swift*, para Android se utiliza *Java*, y para Windows Phone se utiliza *.Net* [15].

Algunas de sus ventajas e inconvenientes son las siguientes:

- **Ventajas:**

  - Tienen un mejor rendimiento debido a que se aprovechan al máximo todos los recursos que permite el *Hardware*.
  - Existe una mejor experiencia del usuario.
  - Constancia en las posibles actualizaciones de las aplicaciones.
  - Es posible que tengan más visibilidad en las plataformas de tienda.

- **Inconvenientes:**

  - Tanto el desarrollo como el mantenimiento tiene un coste bastante mayor.
  - Al ser el código exclusivo para cada sistema operativo, no se puede reutilizar código.
  - Para cada plataforma, hay que desarrollar una solución diferente y la manera de abordar la funcionalidad es distinta.

2.4.2. Aplicaciones móviles Híbridas Web

Las aplicaciones móviles Web [16] son aquellas que su desarrollo es independiente a cualquier sistema operativo, es decir, son capaces de ejecutarse en las diferentes plataformas independientemente, sin tener la necesidad de crear diferentes aplicaciones. En su desarrollo, se utilizan lenguajes como HTML, JavaScript y CSS.

Tienen una base nativa, pero se ejecutan en el propio navegador de cada dispositivo embebido en la base. A continuación, se detallan algunas de las ventajas e inconvenientes que conllevan:

- **Ventajas:**

  - El código desarrollado es reutilizable para las distintas plataformas.
  - Tiene un bajo coste tanto de aprendizaje como de desarrollo.
  - Es posible su utilización en sitios "*responsive*.º auto-ajustable a tamaño de pantalla.

- **Inconvenientes:**

  - Existe un acceso limitado a los recursos.
  - La experiencia del usuario es peor debido a un mayor tiempo de respuesta.
  - La actualización de la base para la gestión de nuevos dispositivos.
  - Implementar posibles controles avanzados no soportados.

Uno de los entornos más conocidos de este tipo de aplicaciones es `Ionic`, una evolución de `Cordova` y `PhoneGap` [15].

2.4.3. Aplicaciones móviles Híbridas Nativas o Cross-Platform Native

Las aplicaciones móviles Híbridas Nativas [17] son las que combinan ciertos aspectos tanto de las nativas como de las web obteniendo algunas de las ventajas de cada una, pero sin tener todas. Es decir, las aplicaciones híbridas pueden lograr un alto rendimiento obteniendo un gran acceso a los recursos hardware que están disponibles, así como el uso de un único código base permitiendo que la solución sea multiplataforma. El resultado obtenido, por tanto, se puede

utilizar en diferentes plataformas cumpliendo una misma funcionalidad y con una interfaz muy semejante [15].

Actualmente, es una de las opciones más escogidas en referencia a las aplicaciones móviles debido a la gran reducción de costes que supone con respecto a las demás opciones, además del progreso que tiene en su aprendizaje. Esto se debe a que usan un lenguaje de alto nivel, que se compila y enlaza con una biblioteca.

Por todo lo expuesto, se ha decido usar este tipo de tecnología para la creación de la aplicación. Estos son algunos ejemplos de herramientas para generar aplicaciones híbridas:

- React Native
- Titanium
- Xamarin

En este caso, se ha optado por el uso de `Xamarin` para el desarrollo de la aplicación, debido a la experiencia en la empresa de otros proyectos y su progresiva mejora.

2.5 Aplicaciones similares a la propuesta

Estas aplicaciones no están comercializadas actualmente, lo que supone que no están entre las millones de aplicaciones que existen en las diferentes tiendas de distribución o en todas las aplicaciones web en Internet. En este tipo de aplicaciones, se ha podido ver que existen una gran variedad de páginas web que ayudan a la gestión del registro, pero no hay una gran variedad en lo que a aplicaciones móviles se refiere.

Por ello, se destacan algunas de las aplicaciones web y móvil usadas para dicho fin y con una funcionalidad similar a estas aplicaciones planteadas y que abarcan este proyecto [18].

- **Mi registro laboral**: es una aplicación móvil que tiene opción para Android e iOS. Permite registrar las horas, firmarlas mensualmente, crear calendarios laborales, añadir bajas y vacaciones, entre otros.
- **Factorial**: es una aplicación web que también tiene su disponibilidad para móvil. Es posible realizar el control horario y cuenta con gestor de ausencias de todo tipo, y, como novedad, de nóminas y contratos para la dirección.
- **Gespymes**: es una aplicación web de gestión de facturación y tesorería, que tiene una opción para llevar el control horario. Es una aplicación un poco tosca para el registro de horas exclusivamente.
- **Geofirma**: es una aplicación móvil multiplataforma que nos permite firmar a través de un código PIN desde una tablet u ordenador, además de permitir registrar las horas por NFC, y un fichaje automático por geolocalización.
- **Work&Track Mobile**: es una solución tanto móvil como web que tiene la posibilidad de llevar el control horario, pero también la gestión de tareas que se realizan.

- **Blaumark**: en este caso nos encontramos que esta aplicación permite registrar las horas desde cualquier dispositivo conectado a Internet, registrando la IP, las coordenadas o la hora del servidor. Su propuesta de seguridad es el uso de blockchain.
- **iAccesos**: es una solución que se puede acceder a través tanto de móvil como web. Permite llevar un control horario, así como generar informes.
- **VisualTime One**: es una solución que permite, además de llevar el control horario, también llevar un control de rendimiento.
- **Presentium**: es una aplicación que ofrece la posibilidad de combinar un lector biométrico y un software para poder fichar, es decir, a través de una huella en combinación con la propia aplicación.
- **CheckJC**: esta solución web que genera unos informes con un aspecto visual muy bueno. Además consigue utilizar tecnologías desde códigos QR hasta NFC para realizar el registro.

3 Análisis

3.1 Introducción

Un proyecto de ingeniería de software siempre debe de seguir una serie de etapas para que el desarrollo concluya con éxito. La primera de ellas y, una de las más críticas e importantes, es el análisis [19].

Se debe entender por completo el problema, estudiar los objetivos y establecer los requisitos que ayuden a definir las características y funciones que va a tener el producto final. Es crucial tener una buena descripción del producto antes de que empiece cualquier trabajo técnico para así poder evitar cambios y posibles retrasos una vez que ya ha dado comienzo.

3.2 Alcance de las aplicaciones

Tracker App, nombre dado al proyecto, son dos aplicaciones (Web y App móvil multiplataforma) para la realización del registro de horas de la empresa de una manera sencilla y con mayor seguridad ya sea, a través de las aplicaciones o un módulo externo (lector de NFC y/o DNIe).

Las aplicaciones surgen con el objetivo de satisfacer las necesidades creadas a raíz de la aprobación de la nueva normativa sobre el registro de horas laborales [Real Decreto-ley 8/2019]. Estos nuevos programas permiten simplificar y mejorar el modelo actual que es utilizado en la empresa, siendo este muy rudimentario y poco fiable ya que requiere de una hoja de cálculo fácilmente manipulable y sin ninguna certificación, fuente potencial de problemas a la hora de garantizar la seguridad de acceso y la automatización del proceso, así como el control de datos.

Esta solución fue planteada para administrar y gestionar internamente en Delonia. Esta versión de las aplicaciones está orientada al acceso web y/o a dispositivos móviles multiplataforma (*Android* e *iOS*). En un futuro se han contemplado posibles ampliaciones de módulos externos para fichar tales como lectores de huella, tarjetas NFC, Beacons, etc.

3.3 Roles

Antes de empezar a definir y plantear los requisitos que debe cumplir el proyecto se deben establecer los distintos roles que van a desempeñar los usuarios dentro del sistema. Van a ser tres roles:

- **Empleado**: es el usuario principal de las aplicaciones. Registra horas y utiliza la aplicación.
- **Administrador**: puede dar de alta o de baja usuarios y solucionar posibles problemas o equivocaciones con los registros. Puede ser un usuario de la empresa o uno externo.
- **Gestor**: se encarga de la instalación e integración de las aplicaciones, así como el mantenimiento de módulos externos.

3.4 Análisis de requisitos

El análisis de requisitos es una etapa crítica tanto para el equipo de desarrollo como para el cliente final. Esto se debe a que unos requisitos que sean incompletos o ambiguos pueden llegar a provocar graves problemas en etapas futuras del proyecto. Por estos motivos, es necesario establecer una buena comunicación con el cliente para evitar este tipo de situaciones.

Por otro lado, es importante tener bien definido el proyecto para poder decidir la viabilidad de éste con una mayor información. Pudiendo así hacer un presupuesto y estimaciones con la mayor precisión posible.

A continuación, están contemplados los requisitos que se contemplan en todo el proyecto. No se han realizado todos los especificados, pero si se han contemplado para hacer un mejor diseño inicial.

En el **Anexo A** se presenta un diagrama de casos de uso en el que se ha basado para definir los requisitos.

3.4.1. Requisitos Funcionales

- **Subsistema de Autenticación**
  - **RF01**: la aplicación permitirá, mediante un usuario y una contraseña, poder acceder a todos las funcionalidades tanto a la aplicación móvil como a la web a usuarios registrados. (App móvil, Web Administradores y Web Empleados)
  - **RF02**: la aplicación permitirá a administradores crear y registrar empleados asignándoles tipo de jornada (completa, media, becario, etc.), correo electrónico y centro de trabajo habitual. (Web Administradores)
- **Subsistema de Registro**
  - **RF03**: el usuario podrá registrar la hora de entrada y salida al trabajo. En él, pondrá el computo global de las horas que ha realizado el usuario. (App móvil y Web Empleados)

- **RF04**: la aplicación permitirá al usuario registrar el tiempo de pausa y/o descansos para las comidas o por algún otro motivo. Estas horas se usarán para el cómputo diario de las horas. (App móvil y Web Empleados)
- **RF05**: el usuario podrá crear incidencias por motivos de ausencia de trabajo por enfermedad u otras causas. Permitirá elegir el día y describir la causa de la incidencia, así como el tipo de ella. (App móvil y Web Empleados)
- **RF06**: la aplicación tendrá un área para poder añadir tanto días de vacaciones por separado, como períodos de tiempo. Se indicará cuantos días de vacaciones le restan y se permitirá una pequeña descripción del motivo. (App móvil y Web Empleados)
- **RF07**: el usuario podrá registrar todos los datos relacionados con las horas y con las ausencias y vacaciones en caso de fallo. (App móvil y Web Empleados)

- **Subsistema de Validación**
  - **RF08**: el administrador podrá ver las horas registradas y pendientes de validación y poder validar las horas que se computan para certificar que son veraces.(Web Administradores)
  - **RF09**: el administrador podrá validar las vacaciones, ausencias y bajas, así como ver los motivos y poder hacer comentarios.(Web Administradores)
- **Subsistema de Modificaciones**
  - **RF10**: el administrador puede crear una solicitud de modificación de horarios por motivos de error y/o algún desajuste.(Web Administradores)
  - **RF11**: el empleado podrá aceptar o denegar modificaciones de los horarios propuestas por un administrador.(App móvil y Web Empleados)
  - **RF12**: la aplicación permitirá a un administrador solicitar horas extra a un empleado. (Web Administradores)
  - **RF13**: la aplicación permitirá a un empleado a aceptar o denegar solicitudes de horas extra. (App móvil y Web Empleados)
- **Subsistema de Localizaciones**
  - **RF14**: la aplicación permitirá crear centros de trabajo y definir su nombre, ubicación y localidad. (Web Administradores)
  - **RF15**: el usuario podrá elegir en que centro de trabajo está fichando debido a posibles cambios de centro. (App móvil y Web Empleados)
  - **RF16**: un administrador podrá definir y establecer calendarios laborales en función de su localización, así como añadir días no lectivos o días con algún comportamiento especial. (Web Administradores)
- **Subsistema de Consulta/Seguimiento**
  - **RF17**: el sistema hará un seguimiento de las horas extra de cada empleado para su control y posible aviso en caso de que no sean acordadas. (App móvil, Web Administradores y Web Empleados)
  - **RF18**: el usuario podrá revisar su calendario, en el se podrá acceder al detalle de cada día y poder hacer algún comentario o modificación de las horas computadas. Estas tendrán que ser validadas. (App móvil, Web Administradores y Web Empleados)

  - **RF19**: la aplicación mostrará estadísticas diarias, semanales y mensuales, de las horas trabajadas, de descanso, horas extra y de vacaciones, así como estadísticas de sus horas a los proyectos. (App móvil, Web Administradores y Web Empleados)
  - **RF20**: el usuario podrá acceder a sus datos de usuario y poder solicitar el cambio de alguno de ellos si se desea. También permitirá añadir una foto de perfil. (App móvil y Web Empleados)

- **Subsistema de informes**
  - **RF21**: el sistema permitirá generar informes de diferentes tipos en función de las necesidades del usuario. (Web Administradores)
  - **RF22**: el usuario, como administrador, podrá descargarse los informes mensuales de todos los datos y las horas, por usuario para su firma y posterior archivo. Contendrá las horas computadas, las vacaciones, las pausas y bajas o ausencias. (Web Administradores)

- **Subsistema de Controladores Externos**
  - **RF23**: el sistema permitirá utilizar algún módulo para poder registrar la hora de entrada y salida fichando (lector de huella, beacon, NFC, etc). (App móvil y Web Empleados)
  - **RF24**: la aplicación permitirá firmar a los empleados sus horas diarias para así certificar que se han computado correctamente. (App móvil, Web Administradores y Web Empleados)
  - **RF25**: la aplicación móvil permitirá el acceso por lector biométrico (huella, reconocimiento facial, etc) siempre que sea posible. (App móvil)

- **Subsistema de Listados**
  - **RF26**: la aplicación mostrará un listado de todas las solicitudes que se hayan realizado con respecto a las vacaciones de cada usuario. (Web Administradores)
  - **RF27**: la aplicación mostrará un listado de todas las solicitudes que se hayan realizado con respecto a modificaciones horarias de cada usuario. (App móvil, Web Administradores y Web Empleados)
  - **RF28**: la aplicación mostrará un listado de todas las solicitudes que se hayan realizado con respecto a las horas extra de cada usuario. (App móvil y Web Empleados)
  - **RF29**: la aplicación mostrará un listado de todos los registros hechos en el día aún sin confirmar. (App móvil y Web Empleados)
  - **RF30**: la aplicación mostrará un listado con todos los registros horarios con necesidad de validación aun pendientes. (Web Administradores)

3.4.2. Requisitos No Funcionales

- **RNF01**: para las peticiones se utilizará un token de tipo JWT generado para cada usuario con el fin de garantizar la seguridad e integridad de las peticiones que se hagan a los servicios.
- **RNF02**: en todas las aplicaciones se seguirá una arquitectura de tipo REST para la creación de servicios.

- **RNF03**: todas las operaciones con los servicios serán de tipo POST.
- **RNF04**: no será expuesta ningún tipo de información sensible, así como guardada en posibles sitios vulnerables en las diferentes aplicaciones.
- **RNF05**: se cumplirá de manera estricta la nueva ley europea con respecto a la protección de datos.
- **RNF06**: el registro de horas se podrá hacer en menos de 5 pasos o 5 minutos.
- **RNF07**: el usuario tardará menos de 1 minuto en acceder a los contenidos de la aplicación.
- **RNF08**: la aplicación tardará menos de 2 segundos a la hora de cargar respuestas.
- **RNF09**: la aplicación no se bloqueará cuando este realizando operaciones.
- **RNF10**: aparecerá un icono de cargando cuando se realicen operaciones internas en las aplicaciones y se demoren demasiado.
- **RNF11**: la aplicación móvil tendrá compatibilidad total con Android 6.0 (API 23) o superior.
- **RNF12**: la aplicación móvil tendrá compatibilidad total con iOS 7.0 o superior.
- **RNF13**: las aplicaciones Web tendrán compatibilidad con todos los buscadores Web más importantes (Chrome, Mozilla, Microsoft Edge, Opera, Internet Explorer...).
- **RNF14**: las aplicaciones deberán adaptarse a todas las resoluciones de pantalla.
- **RNF15**: la aplicación guardará registros de todas las acciones que realice el usuario.

3.5 Solución propuesta

En esta primera versión y, por tanto, lo que se ha podido abarcar en este TFG han sido los siguientes subsistemas:

- Subsistema de Autenticación.
- Subsistema de Registro.
- Subsistema de Modificaciones.
- Subsistema de Localizaciones.
- Subsistema de Consulta.

También añadir que, la Web de Administradores no se ha contemplado en esta entrega y, los requisitos asociados, no se han introducido en esa aplicación.

El motivo por el que no se realizan todos los requisitos contemplados se debe a que es un proyecto muy ambicioso, pero su análisis completo ayuda a un mejor diseño de las aplicaciones.

4

Diseño y Arquitectura

4.1 Introducción

El diseño de la aplicación empieza una vez que ya se han definido todos los requisitos de la fase de análisis. En esta etapa se va a definir la arquitectura a utilizar en el *front-end* o capa de presentación, el *back-end* o capa de acceso de datos y en la base de datos [20]. También se definen los servicios a implementar para conectarnos con el servidor y los diseños de las maquetas para la interfaz gráfica.

Por tanto, consiste en definir y diseñar todos los componentes del sistema respondiendo a las funcionalidades descritas en el análisis. Además, se hablará de las diferentes herramientas que se han usado para la realización de los diseños.

4.2 Patrón de diseño

La patrón de diseño utilizado tanto en la aplicación móvil como en la aplicación web ha sido el Modelo-Vista-VistaModelo (MVVM). Este patrón es un derivado del patrón Modelo-Vista-Controlador (MVC), pero su principal diferencia es que en el MVVM, la Vista toma un papel más importante que en MVC [21].

En MVC se separan los datos de la aplicación, la interfaz de usuario y la lógica de negocio en tres partes totalmente diferenciadas. Esto produce que cuando se realiza un cambio en la lógica, es ella quien tiene que actualizar la Vista, es decir, es el Controlador el que decide las interacciones entre la Vista (interfaz) y el Modelo, por tanto, es el que primero y último en actuar [22].

Por otro lado, el patrón MVVM, del mismo modo que tiene MVC, cuenta con un Modelo en el cual se encuentran los datos y la información, pero no contiene las acciones que se realizan. También contiene con una Vista que se encarga de mostrar todos los datos y pantallas, pero,

en este caso, esta es activa, es decir, puede poseer acciones y enlazar datos con lo que así tiene algún conocimiento del Modelo de datos que existe. Finalmente, el Modelo-Vista se considera un intermediario entre la Vista y el Modelo [23]. El comportamiento del este patrón se puede ver en la Figura 4.1.

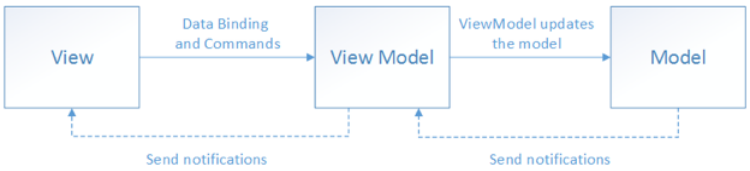

Figura 4.1: Esquema MVVM.

Fuente: docs.microsoft.com

4.3 Arquitectura

La arquitectura en la que se basan las dos aplicaciones (Web y móvil) es la estructura básica que se utiliza en diseño de software. Consta de una capa de presentación (*front-end*) que está formada por la parte del software que interacciona con los usuarios, y una capa de acceso a datos (*back-end*) que se ocupa de procesar las peticiones que le llegan del usuario realizando las operaciones necesarias en la base de datos.

El *back-end* va a constar de una base de datos relacional SQL y un servidor desarrollado en Java junto con un *framework* propio de la empresa Delonia llamado `Uaithne`. Este entorno, evita la repetición de algunas partes de código al implementar el servidor, así como el seguimiento de un estándar a la hora de estructurarlo.

Las comunicaciones entre *front-end* y *back-end* se basarán en una conexión HTTPS con una arquitectura de peticiones REST. Además, estas conexiones necesitarán un *Authenticacion Token* el cual será obtenido a través de una herramienta que permite el inicio de sesión único llamada `Keycloak`.

Este token caducará después de un tiempo determinado así haciendo necesario que se vuelva a iniciar sesión. Además irá codificado con `JWT` (Json Web Token). Todos lo datos que se envíen o reciban irán en formato JSON.

Además, en los servicios que se implementan, se va a seguir un estándar de descripción para API's de tipo REST llamado `Open API` [24]. Este estándar es una interfaz *OpenSource* que proporciona un acceso de manera programática a un servicio web, es decir, que se accede siempre de una misma manera, simplificando el acceso.

4.4 Base de datos

La base de datos requiere también un diseño previo a empezar con el desarrollo. Un buen diseño podría facilitar las operaciones posteriores, reduciendo así posibles problemas futuros, coste y tiempo.

Además, va a ser una base de datos relacional SQL y sigue una estructura que está en 3NF. En el **Anexo A**, se puede ver el diseño de la base de datos a partir de un diagrama ER.

4.5 Maquetas

El proceso de maquetación permite mostrar una interfaz de la aplicación al cliente siguiendo los requisitos especificados con el objetivo de obtener una idea más clara del producto que se va a realizar.

Así, los cambios de diseño que se produzcan en esta etapa ocasionará problemas menores de los que supone un cambio de diseño de la interfaz cuando ya está en desarrollo. Por este motivo, es de especial importancia crear todas las versiones que se necesiten, compararlas y definir el diseño final para evitar problemas futuros. Además de diseñar las pantallas, es muy importante mostrar el flujo que van a tener todas las pantallas de la aplicación ya que facilita el trabajo de desarrollo y muestra mejor el comportamiento de la aplicación.

En el **Anexo C** se han incluido ejemplos los diseños realizados de las dos aplicaciones. Son diseños prototipo y se revisarán con una persona de diseño para mejorarlo y hacerlo más atractivo visualmente.

4.6 Análisis de herramientas utilizadas

En esta sección, se expondrán y explicarán las herramientas utilizadas para la etapa de diseño y la arquitectura de las aplicaciones.

4.6.1. Visual Paradigm

La herramienta utilizada para el diseño de la base de datos ha sido Visual Paradigm [25] debido a que, además de ser una herramienta que permite tanto el diseño de las bases de datos como cualquier tipo de diagramas UML, tienes la posibilidad de crear un *script* con el código SQL que te genera la base de datos.

Para que la base de datos sea consistente, es necesario que el diseño de la misma sea correcto, indicando las relaciones entre las diferentes tablas así como sus claves primarias y foráneas.

4.6.2. Adobe XD

En lo que se refiere al diseño de la interfaz gráfica se ha decidido usar Adobe XD [26]. Es una herramienta que pertenece a Adobe que permite generar maquetas con muchos detalles y que ayudan al desarrollo final de la aplicación incluyendo muchos componentes y paquetes de interfaces de usuario para todo tipo de aplicaciones. También, facilita el desarrollo debido a que permite ver los márgenes, los colores, el tipo y tamaño de letra, además de ver el flujo de las pantallas de la aplicación.

4.6.3. Apicurio Studio

La herramienta elegida para el diseño de los servicios que se van a implementar ha sido Apicurio Studio [27]. Se ha decidido así debido a que es *open source* con gran cantidad de opciones diferentes para poder diseñar una API. Además, sigue el estándar Open API 3.0 el cual describe un formato a seguir para las API's de tipo REST con anotaciones para todas las operaciones.

Cuando ya se han definido los servicios que se van necesitar, es posible generar la documentación necesaria incluyendo el protocolo a usar, si es necesario un token de verificación y los distintos valores de entrada y salida de cada petición.

Como sigue un estándar, esto nos permite generar ejemplos de los servicios para, posteriormente, poder generar un archivo YAML. Finalmente y, con la ayuda de otra herramienta llamada APISprout, es posible simular las respuestas que nos daría el servidor real.

También se ha podido generar un archivo JSON que, en conjunto con el plugin de Maven *open-api-generator* y el generador *resteasy*, se ha podido generar la API para la implementación del servidor.

5
Desarrollo

5.1 Introducción

Al terminar las etapas de Análisis y Diseño del proyecto es cuando se puede empezar la etapa de Desarrollo de la aplicación. Un gran esfuerzo y trabajo en las etapas anteriores puede reducir considerablemente el tiempo invertido para el desarrollo del proyecto, que es la etapa más larga de todas.

El desarrollo del proyecto se ha iniciado con el desarrollo del *front-end* empezando por la aplicación móvil y siguiend, a continuación, con la web. Posteriormente, se desarrollaron los servicios web que se iban a usar y se creó la base de datos. Esta se instancia localmente para agilizar el desarrollo. Finalmente, se desarrolló el *back-end* para así poder hacer las distintas operaciones en la base de datos.

5.2 Front-End

En el desarrollo de la capa de presentación se ha utilizado, por un lado, `.Net` junto con Xamarin.Forms como entorno de trabajo y Visual Studio como entorno de desarrollo para crear aplicaciones móvil multiplataforma y, por otro, `JavaScript` junto con React como *framework* y Visual Studio Code como editor de código para el desarrollo de la aplicación web.

5.2.1. Visual Studio IDE

Visual Studio IDE [28] es un entorno de desarrollo integrado de Microsoft. Su lanzamiento oficial fue en 1997 pero fue en 2002 cuando añadió la funcionalidad para crear aplicaciones móviles, aplicaciones web y otro tipo de aplicaciones para entornos que soporten .NET.

El entorno soporta gran cantidad de lenguajes de programación tales como C#, Java, Ruby,

Python, F#, PHP, etc. Además, también soporta entornos de desarrollo como Django y es compatible tanto con Windows como con Mac (a partir de la versión 2017).

Actualmente está en funcionamiento la versión Visual Studio 2019 y es la que se ha utilizado para el desarrollo debido a que es compatible con Xamarin Forms. Anteriormente, se necesitaba Xamarin Studio para desarrollarlo, pero en 2017 se integro en Viaual Studio coincidiendo con el lanzamiento de la plataforma para Mac.

Esta herramienta, por tanto, se ha utilizado para el desarrollo de la capa de presentación de la aplicación móvil, utilizando Xamarin Forms y distintos *plug-in's* desarrollados por Microsoft y la comunidad de Xamarin Forms.

5.2.2. Xamarin.Forms

Xamarin.Forms [29] es una característica de Xamarin, un *framework* de código abierto para el desarrollo para aplicaciones móviles multiplataforma (iOS, Android y Windows Phone) sobre un mismo código base. Extiende de la plataforma de .NET que cuenta con librerías y herramientas para crear dichas aplicaciones [30].

El hecho de haber elegido Xamarin ante otros entornos similares como React Native se ha debido, a que, en la empresa, estaba implantado desde hace ya algunos años, estando su elección del uso de Xamarin.Forms basada en un trabajo de fin de grado para la UAM [31] que hizo un compañero de la empresa sobre todos los entornos de desarrollo posibles para el desarrollo de las aplicaciones multiplataforma y sus comparativas. Este estudio se realizó previamente a la compra de Xamarin por Microsoft.

Una de las características principales que tiene C# es la sencillez para poder desarrollar tareas asíncronas, un hecho muy importante en la creación de aplicaciones móviles porque de ésta manera conseguiremos no sobrecargar el hilo principal. El motivo es que la aplicación se encuentra en ejecución en dicho hilo pudiendo llegar incluso a bloquearse y cerrarse si se sobrecarga en exceso. En lo que respecta a su programación, en C# destaca su sencillez a la hora de trabajar con ello, así con llamadas a los métodos *await* en conjunto con *async* y la palabra *Task* conseguiremos crear dichas tareas. En el **Anexo B** se exponen unos ejemplos de este desarrollo, además de algún ejemplo para añadir componentes gráficos en C#.

Por último, otra de las ventajas con las que cuenta Xamarin.Forms es el uso de los llamados *binders*. Consiguen enlazar dos objetos, pudiendo así, en el caso de que sepamos que un objeto en la aplicación vaya a variar, pero no sepamos que valores va a tener se puede utilizar un *binder* para así unir los dos objetos y que se actualice cada vez que cambie. Su uso es muy común en el patrón MVVM que hemos utilizado en el proyecto. En el **Anexo B** se puede encontrar un ejemplo de como es su programación.

5.2.3. React

React.js [32] es una librería en Javascript de código abierto que fue creada por Facebook para facilitar el desarrollo de interfaces de usuario de aplicaciones en una sola página. El objetivo principal de React es ser sencillo, declarativo y fácil de combinar ayudando así a los programadores a crear aplicaciones que utilizan datos que varían constantemente [33].

React.js solo proporciona la Vista en un contexto como el de esta aplicación que se utiliza

un patrón MVVM. Esta Vista se basa en la creación de componentes interactivos y reutilizables que capturan las actualizaciones que va teniendo la página actual y, que se traducen en una muestra virtual de la página resultante. Es decir, cuando React detecta un cambio en su estado, se ejecutan nuevamente las funciones para poder crear una nueva representación. Esto lo hace a partir de un algoritmo que determina las diferencias entre la página virtual y la actual realizando los cambios necesarios del DOM para reflejar la nueva representación. Para que React sea más eficiente que el uso de JavaScript habitual utiliza un concepto llamado DOM virtual el cual realiza selectivamente sub-árboles de los nodos sobre los cambios de estado con la menor cantidad de manipulación en el DOM [34].

Una de sus particularidades, es el hecho de como se crean los componentes que posteriormente se usan en las interfaces, así como los métodos y la actualización de ellos. En el **Anexo B** se puede ver un ejemplo de componente en React que en esencia se compone de un elemento de estado de la página actual llamado *state* y de las propiedades heredadas de la página padre llamadas *props*. Estos dos elementos son los que van guardando la información entre páginas y actualizando el estado de cada una de ellas. Además, también se ha añadido la clase que corresponde al control del cambio entre las distintas páginas.

Finalmente, la utilización de esta herramienta en lugar de usar otros *frameworks* como Angular.js ha sido, en gran medida, a su creciente expansión y estado de auge debido a que solo se encarga de la gestión de la vista pudiendo así adaptarse a otros entornos. Otro de los motivos ha sido su utilización en otros proyectos de la empresa pudiendo así poder tener ciertas referencias y un apoyo mayor en el caso de que surgieran problemas.

5.2.4. Visual Studio Code

Visual Studio Code [35] es un editor de código fuente desarrollado por Microsoft de código abierto y multiplataforma. Incluye soporte para el control de la depuración, control de versiones, resaltado de la sintaxis, etc.

Se ha escogido este editor para desarrollar todo el código en React debido a que se basa en Electron, que es un entorno de desarrollo que se usa para desarrollar aplicaciones Node.js [36]. Además, también se ha decidido usar debido a que se añaden de manera muy sencilla librerías que se utilizan para comprobar la sintaxis del código y, por tanto, evitar problemas a la hora de compilar la aplicación.

Figura 5.1: Logo de Visual Studio Code.
Fuente: launchdarkly.com

5.3 Base de datos

En la gestión y administración para la base de datos se ha utilizado PosgreSQL como gestor de la base de datos relacional y SQLDeveloper como administrador.

5.3.1. PostgreSQL y SQLDeveloper

En primer lugar, el gestor de base de datos que se ha utilizado para la base de datos relacional ha sido PostgreSQL [37]. Es un proyecto de código abierto y está orientado a objetos. Entre sus características principales están, la alta concurrencia que permite varios accesos simultáneamente sin bloqueos y la amplia variedad de tipos nativos que soporta [38].

Por otro lado, para administrar la base de datos se ha utilizado SQLDeveloper [39]. Es una interfaz gráfica de usuario gratuita propiedad de Oracle que permite a los usuarios tener acceso a potentes editores para trabajar con SQL, ejecutar consultas y modificar datos de una manera fácil e intuitiva. También tiene varías herramientas tales como que proporciona a los administradores un conjunto de interfaces para las tareas más críticas o una solución completa para el modelado de datos.

5.4 Servicios Web

Como se ha planteado en el diseño, las distintas comunicaciones entre el servidor y el *front-end* se realizan a través del protocolo HTTPS garantizando de esta manera el envío seguro de los datos ya que van cifrados. La arquitectura de servicios es de tipo REST y en conjunto con HTTPS se ha utilizado el *token* de autenticación JWT.

Finalmente, para el inicio de sesión se ha utilizado un producto de software, llamado Keycloak, para la gestión y administración de los distintos usuarios.

5.4.1. KeyCloak

Keycloak [40] es una solución de gestión de acceso e identidad de código abierto dirigida a aplicaciones y servicios modernos. Su utilidad es la de facilitar el inicio de sesión a los usuarios, proporcionando un servicio único que soporta diferentes estándares, además de hacer las aplicaciones más seguras y su servicio más simplificado. Los usuarios se registran y es el propio entorno el que se ocupa de su gestión en las distintas aplicaciones.

Una de sus principales características es que tiene inicio de sesión único. Esto quiere decir que todas las aplicaciones que se gestionen por Keycloak se autentican por su *login* único y una vez iniciada sesión, los usuarios ya no tienen que acceder nuevamente a la aplicación. Además, permite habilitar el inicio de sesión con redes sociales debido a que permite autenticar usuarios con los proveedores de identidad OpenID Connect, OAuth 2.0 o SAML 2.0.

Finalmente, en la gestión de usuarios incluye un sistema que genera un *token autentication* para garantizar la confidencialidad e integridad de los servicios que usen este inicio de sesión. Este token se puede configurar y en esta aplicación se ha usado JWT, un estándar del que se hablará en el apartado siguiente [41].

Como se puede ver, se ha decidido usar esta solución, en vez de administrar los usuarios con código hecho a medida, debido a que simplifica mucho todos los procesos de creación y gestión de cada usuario, y evita la generación manual de los *tokens* JWT.

5.4.2. JWT

Como ya se ha explicado anteriormente, utilizando el protocolo HTTPS en conjunto con la arquitectura REST hemos conseguido garantizar integridad y confidencialidad en los servicios, pero para lograr la verificación de la información se ha usado JWT [42]. Éste es un estándar que nos permite utilizar un *token authentication* para poder llegar a obtener esa integridad del mensaje.

JWT se estructuran en tres fracciones diferenciadas (ver Figura 5.2) que son el encabezado, los datos y la firma todos ellos separados por un punto. El encabezado muestra el algoritmo que se ha utilizado para crear la firma, en este caso RSA256, generando una clave pública y una clave privada generada y almacenada por Keycloak. Los datos son la parte del contenido, es decir, es donde se guarda todo la información que guarda el *token* como, por ejemplo, identificador del usuario, email, roles, etc). Por último, está la firma que se establece al cifrar el encabezado y los datos [43].

Finalmente, el *token* se obtiene cuando se ha hecho sesión en Keycloak a través de la aplicación y se almacena en la cabecera de las peticiones. De este modo, le sirve al usuario para poder acceder a los demás servicios que contenga la aplicación por un tiempo determinado, ya que tiene una marca temporal que muestra el tiempo que está activo. Pasado ese tiempo, caducará la sesión y se tendrá que volver a iniciar sesión.

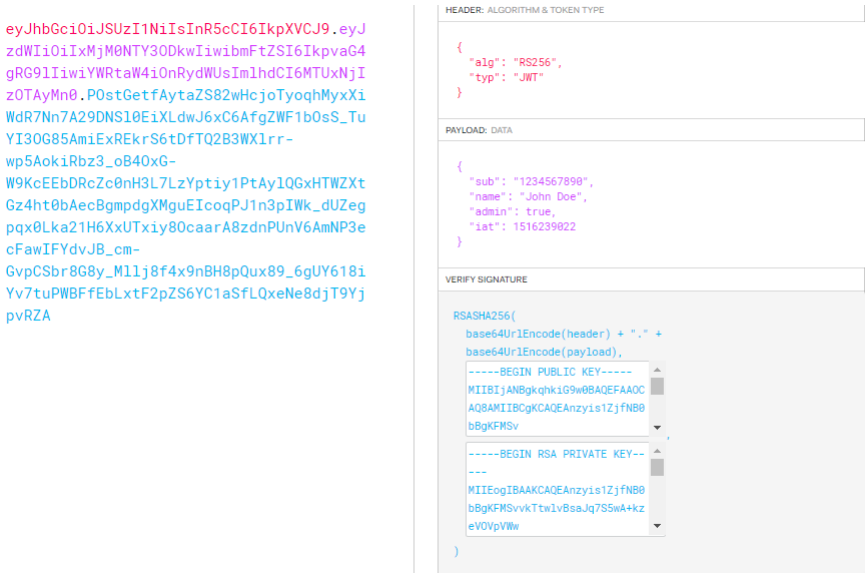

Figura 5.2: Ejemplo de un token JWT.
Fuente: jwt.io

5.4.3. APISprout

APISprout [44] es un servidor que sirve para simular las respuestas que va a producir una determinada API. De esta forma y, debido a que sigue el estándar OpenAPI, podemos simular nuestro servidor a través de un archivo YAML. Los servicios se han diseñado con la herramienta Apicurio que, al seguir el mismo estándar y generar este archivo, permite definir ejemplos y generar documentos con las especificaciones de los servicios.

Esto posibilita comprobar que los servicios se han realizado de la manera correcta antes de comenzar el desarrollo del *back-end*. Además, es posible el desarrollo de otros módulos y su correcto funcionamiento, como el *front-end*, sin depender de que se hayan implementado ya los servicios.

5.5 Back-End

El *back-end* o servidor es el intermediario que se ocupa de implementar la lógica de negocio y persistir la información en la BBDD.

Para desarrollar el servidor se ha utilizado el IDE Netbeans y Java como lenguaje de desarrollo. Para el acceso de los datos se ha utilizado Uaithne, un *framework* desarrollado en Delonia, que facilita la construcción de los objetos que permiten la consulta, creación y actualización de los datos. Por último, se ha desplegado el servidor en Tomcat 8.5.

5.5.1. NetBeans

Netbeans IDE [45] es un entorno de desarrollo integrado libre, de código abierto y gratuito sin ninguna restricción para su uso. Fue fundado por Sun MicroSystems en el año 2000 y aún sigue siendo el patrocinador principal de los proyectos.También, garantiza y permite el uso de una gran variedad de tecnologías para el desarrollo de aplicaciones Web en los que se incluyen tipos de proyectos como Maven, aplicaciones para escritorio o aplicaciones móviles. Además, también soporta muchos lenguajes de programación como, por ejemplo, Java el cual es el principal del IDE, PHP, Groovy, C/C++, HTML, CSS, etc.

En lo que se refiere a sus características, una de las principales y, por lo que se ha utilizado en este proyecto, es que se integra con diversidad de servidores de aplicaciones, pudiéndolos gestionar desde el propio entorno: iniciar, parar, arrancar en modo depurador, desplegar, etc. Es compatible con GlassFish, JBoss, WebLogic, Sailfin y, por supuesto, con el que se ha usado en este proyecto, Apache Tomcat. También podemos destacar sus herramientas para el depurado de los errores en el cual podemos definir *breakpoints* o puntos de ruptura para monitorizar en tiempo real los valores de las variables y de las propiedades, así como ver lo que devuelven distintos métodos [46].

5.5.2. Maven

Maven [47] es una herramienta de código abierto que se creó en 2001. Su objetivo era el de simplificar los procesos de compilar y generar los ejecutables a partir de código fuente o también conocido como procesos de *build*.

Anteriormente, los desarrolladores invertían bastante tiempo en aprender las características de cada proyecto debido a que cada uno tenía una manera para configurar su proceso de *build*. En el caso de que se quisiera compilar y generar los distintos ejecutables había que analizar que porciones de código se iban a compilar, que librerías se utilizaban, que dependencias tenían, etc. Por esto, se podía llegar a invertir mucho tiempo. Era un proceso lento, complejo y propenso a errores.

Maven simplifica todo este proceso ya que independientemente de las dependencias, librerías y el código que haya que compilar, consiste en ejecutar una simple instrucción (*mvn intall*). Además, en Maven se definen tres ciclos del proceso *build* con una serie de etapas distintas como la validación, la verificación y el despliegue en los entornos necesarios como integración de desarrollos y pruebas.

Finalmente, Maven permite la gestión de librerías y módulos de manera muy simple. Solo es necesario introducir estos módulos y librerías en un fichero de configuración del proyecto llamado POM. En el caso de este proyecto, se ha configurado en este fichero un generador siguiendo el estándar `OpenAPI` llamado RestEasy para generar las clases y los servicios necesarios para la API [48] [49].

5.5.3. Uaithne

En primer lugar, se explica qué es Uaithne y cuál es su objetivo, debido a que estas son las primeras dudas que suelen surgir citando al creador de esta solución: "*Uaithne es un proyecto que planteo en respuesta a la necesidad de tener una arquitectura sólida en el back-end de las aplicaciones y a la vez conseguir mejorar significativamente la productividad del equipo al desarrollar esta parte de la aplicación. La filosofía central de Uaithne es conseguir que el back-end se asemeje a un juego de LEGO©, en donde se disponen de piezas altamente intercambiables que se componen hasta conseguir el resultado deseado, cuya función final es conseguir la ejecución de una operación (comando)*" [50].

Uaithne surge por la combinación de algunos patrones de diseño como son Visitante (*Visitor*), Orden (*Command*), Cadena de responsabilidad (*Chain of Responsability*), y Estrategia (*Strategy*) [21]. En conjunto, todos estos patrones forman una configuración sobre la que poder crear el servidor utilizando una programación orientada a aspectos. La arquitectura se conforma de 5 componentes principales en los cuales la información y la lógica están claramente separados. Cada componente se explica en la Tabla 5.1.

En lo que se refiere al acceso a BBDD, Uaithne utiliza MyBatis el cual es un *framework* de persistencia que soporta SQL eliminando así casi todo el código en JDBC. Además, tiene anotaciones para la generación de la consulta automáticamente pudiendo variar campos de la consulta si es algo más compleja.

Uaithne fue desarrollado en Delonia y, actualmente, se utiliza en la mayoría de los proyectos que están en activo manteniendo así una misma metodología en la empresa. En un principio se hizo únicamente para Java, pero se ha realizado una adaptación también para *.NET Core* en C# [51].

Por último, Uaithne también funciona como un generador de código ahorrando así al desarrollador programar líneas de código repetidas que surgen cuando se están creando las distintas operaciones y accesos a la base de datos.

Componente	Definición
Entidad (Entity)	Clase que contiene toda la información de una tabla de la BBDD, es decir, todos los campos.
Vista de Entidades (EntityView)	Clase que contiene solo la información que es necesaria de una tabla de la BBDD.
Operaciones (Operation)	Clase que contiene la información necesaria para que el sistema ejecute la acción. Se agrupan en módulos.
Ejecutores (Executor)	Encargados de ejecutar las operaciones de un módulo. Cada uno define su propio ejecutor.
Bus de ejecución (Execution Bus)	Para conectar los distintos módulos y ejecutar la operación se crea el Bus de Ejecución.

Tabla 5.1: Componentes de Uaithne.
Fuente: Manual de Uaithne.

En el **Anexo B** se muestran ejemplos de los distintos componentes así como la notación utilizada.

5.5.4. Tomcat

Apache Tomcat [52] es un contenedor web que soporta servlets y JSP's incluyendo un compilador Jasper. Aunque Tomcat no es un servidor de aplicaciones, como JBoss o JOnAs puede funcionar como servidor web por si mismo. Está escrito en Java y, por tanto, funciona en cualquier sistema operativo que tenga la máquina virtual Java 8.0.

Tomcat fue desarrollado por el proyecto Jakarta en la fundación Apache Software, implementando especificaciones de los servlets. Es un proyecto de código abierto bajo la licencia de Apache [53].

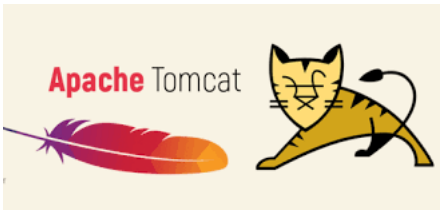

Figura 5.3: Logo Apache Tomcat.
Fuente: unaaldia.hispasec.com

6

Pruebas y Resultados

6.1 Introducción

Cuando se ha terminado la fase de desarrollo se han realizado las pruebas. El objetivo de ellas es el de comprobar que el comportamiento de la aplicación y el sistema cumplen con los requisitos que se establecieron en la fase de análisis.

Estas pruebas también demuestran que el software está libre de defectos. Las pruebas se han realizando al *front-end*, al *back-end* y a los servicios utilizados.

6.2 Pruebas de verificación

En estas pruebas se pasan para comprobar que el software consta de todos los requisitos definidos en la fase del análisis.

6.2.1. Pruebas de Caja Negra

En estas pruebas solo se tienen en cuenta la entrada y la salida sin tener en cuenta la lógica interna.

Tanto para la aplicación móvil como para la web se han realizado estas pruebas usando las aplicaciones. Para conseguir obtener el mayor número posible de pruebas ha colaborado una persona del equipo de QA (*Quality Assurance*) de la empresa y que se ha dedicado, en conjunto conmigo a diseñar y ejecutar el plan de pruebas permitiendo verificar el correcto funcionamiento de las aplicaciones y consiguiendo así obtener el mayor número de flujos posibles. Cada vez que se obtenía un error, se añadía una fila a un Excel en el cual se van registrando todos los errores. Cada uno tiene un identificador, un nombre, una prioridad y una descripción para poder saber donde está el error.

6.2.2. Pruebas de Caja Blanca

En estas pruebas se conoce la lógica de los módulos a probar que suelen ser módulos de tamaño pequeño. Además se busca que se ejecute al menos una vez cada sentencia de código.

Como se ha dicho, se han realizado las pruebas tanto en el *front-end* como en el servidor, aunque en este último de manera más incisiva y haciendo pruebas más exhaustivas en los módulos más complejos.

Cuando un módulo pasaba todas las pruebas, se pasaba al siguiente.

6.3 Pruebas de validación

En estas pruebas se ha comprobado la concordancia directa de las aplicaciones con los requisitos planteados. Se ha ido revisando uno por uno y su funcionalidad para comprobar que todos los requerimientos contemplados se satisfagan de manera correcta.

Al ser un proyecto interno y, haber tenido una buena comunicación con los compañeros en la empresa, se han ido revisando las aplicaciones y los diferentes diseños. Todo esto, en conjunto con el trabajo hecho en la etapa de análisis, los errores que han surgido en las pruebas han sido menores.

7

Conclusiones y Trabajo futuro

Este Trabajo de Fin de Grado surgió como una solución a un problema que Delonia Software pudo ver en el proceso de registro de las horas. Esta solución me la propuso la empresa y, viendo que podía ser una oportunidad para aprender nuevas tecnologías que no conocía además de obtener una experiencia de estar en un proyecto real,lo acepté.

El proyecto ha consistido en la creación de una aplicación móvil multiplataforma y una aplicación web que tiene por objetivo que los empleados de una empresa registren las horas trabajadas y así se pueda llevar un control de asistencia.

En este proyecto, y durante su desarrollo, he adquirido experiencia en cómo funciona un proyecto desde su inicio, cuando se plantea la solución, hasta las pruebas realizadas y los resultados obtenidos. En todas las etapas del proyecto he ido aprendiendo nuevos conocimientos dándome cuenta de la importancia de algunas de las herramientas y conocimientos utilizados en la universidad.

Alguna de las cosas importantes que he aprendido ha sido la realización de pruebas en el desarrollo y el registro de los defectos encontrados para su posterior solución. El usar una hoja de cálculo ha permitido abordar el registro y control de errores. Como mejoras futuras se puede usar una herramienta más adecuada como puede ser Jira [54], donde se pueden implementar *plugins* para documentar el plan de pruebas y también para registrar y controlar las incidencias detectadas de una forma más eficiente y colaborativa.

Por otro lado, también he aprendido la importancia que tiene la elección de las tecnologías en la realización de los proyectos teniendo en cuenta el posible trabajo futuro que se tenga que realizar.

Además, me gustaría destacar que el desarrollo de la memoria se ha realizado con la herramienta *Latex*. Nunca la había utilizado y, con su desarrollo, puedes ver lo potente que puede llegar a ser. Permite elaborar con facilidad los documentos, con un estilo marcado, generando la bibliografía, creando los índices, aplicando los márgenes, etc.

Por último, se presenta una serie de posibles trabajos futuros a realizar:

- **Web para los administradores**: aunque forma parte de los requisitos, no se ha contemplado para esta fase del proyecto debido a la dimensión del mismo y, por tanto, se ha dejado para una fase posterior. Consistiría en crear una web para administrar todas las horas y poder tener un desglose de la empresa además de poder gestionar la base de datos.
- **Uso de módulos externos**: uso de algún módulo externo para poder registrar dichos eventos como, por ejemplo, NFC asociado a una pantalla, lector de DNIe para firmar, etc. No ha sido posible en esta versión debido a la envergadura del proyecto.
- **Soporte a Multilenguaje**: consistiría en añadir unas tablas con los textos en otros idiomas y poder obtener los textos del idioma escogido.

Bibliografía

[1] M. O. Herrera, "Métodos y técnicas para la gestión de proyectos software," Master's thesis, Universidad de Sevilla, España, 2010. [Online]. Available: http://bibing.us.es/proyectos/abreproy/70193/fichero/3.+METODOLOG%C3%8DAS+DE+GESTI%C3%93N+DE+PROYECTOS.pdf

[2] "Metodologías de trabajo. scrum." [Online]. Available: https://www.softeng.es/es-es/empresa/metodologias-de-trabajo/metodologia-scrum/proceso-roles-de-scrum.html

[3] "Sourcetree." [Online]. Available: https://www.sourcetreeapp.com

[4] G. de España, "Real decreto-ley 8/2019." Jefatura Del Estado, Tech. Rep., 2019. [Online]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-2019-3481&p=20191012&tn=1

[5] "Obligatoriedad de registro de la jornada de los trabajadores." [Online]. Available: https://www.iberley.es/temas/obligatoriedad-registro-jornada-trabajadores-61342

[6] Neosoft, "¿qué es una aplicación web?" [Online]. Available: https://www.neosoft.es/blog/que-es-una-aplicacion-web

[7] S. Luján-Mora, "Programación de aplicaciones web: historia, principios básicos y clientes web," 2002-10-31. [Online]. Available: http://hdl.handle.net/10045/16995

[8] P. Skólski, "Single-page application vs. multiple-page application," 2016. [Online]. Available: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58

[9] K. Yan, "An overview of modern web application frameworks," 2019. [Online]. Available: https://bravoka.io/articles/modern-web-application-frameworks

[10] D. J. Guru, "Single page application: Un viaje a las spa a través de angular y javascript." [Online]. Available: https://medium.com/@davidjguru/single-page-application-un-viaje-a-las-spa-a-trav%C3%A9s-de-angular-y-javascript-337a2d18532

[11] A. D., "Single-page app vs multi-page app: Choosing a way to deliver the best user flow." [Online]. Available: https://www.cleveroad.com/blog/single-page-app-vs-multi-page-application-what-to-choose#multi-page-applications-pros-and-cons

[12] J. Clement, "Number of apps available in leading app stores as of 3rd quarter 2019," 2019. [Online]. Available: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores

[13] M. C. Gtz, "Tipos de aplicaciones móviles." 2014. [Online]. Available: https://www.lancetalent.com/blog/tipos-de-aplicaciones-moviles-ventajas-inconvenientes

[14] M. Rouse, "Native app," 2018. [Online]. Available: https://searchsoftwarequality.techtarget.com/definition/native-application-native-app

[15] L. Valdellon, 2019. [Online]. Available: https://clevertap.com/blog/types-of-mobile-apps

[16] Anónimo, "Web app: Qué es, ventajas, características y ejemplos." [Online]. Available: https://www.tu-app.net/blog/web-app

[17] A. Manchanda, "Cross-platform app frameworks in 2019," 2019. [Online]. Available: https://www.netsolutions.com/insights/cross-platform-app-frameworks-in-2019/

[18] C. Roberto, "19 aplicaciones de registro horario recomendadas por los lectores de pymes y autónomos," 2019. [Online]. Available: https://www.pymesyautonomos.com/legalidad/19-aplicaciones-registro-horario-recomendadas-lectores-pymes-autonomos

[19] R. S. Pressman, *Ingenieria del Software. Un enfoque práctico.* Obregón, México: McGrawHill, 2010.

[20] "Front-end y back-end." [Online]. Available: https://devcode.la/blog/frontend-y-backend/

[21] E. Gamma, *Patrones de diseño: Elementos de software orientado a objetos reutilizables.* Pearson Education, 2002.

[22] "Mvc y mvvm," 2012. [Online]. Available: https://www.adictosaltrabajo.com/2012/10/07/zk-mvc-mvvm/

[23] "Patrón model-view-viewmodel," 2017. [Online]. Available: https://docs.microsoft.com/es-es/xamarin/xamarin-forms/enterprise-application-patterns/mvvm#the-mvvm-pattern

[24] "Open api." [Online]. Available: https://swagger.io/docs/specification/about/

[25] "Visual paradigm." [Online]. Available: https://www.visual-paradigm.com

[26] "Adobe xd." [Online]. Available: https://www.adobe.com/es/products/xd/details.html

[27] "Apicurio studio." [Online]. Available: https://www.apicur.io

[28] "Visual studio." [Online]. Available: https://visualstudio.microsoft.com/es/vs

[29] "Xamarin forms." [Online]. Available: https://docs.microsoft.com/es-es/xamarin/xamarin-forms

[30] M. A. Gómez, "Xamarin forms, apps nativas multiplataforma." [Online]. Available: https://softwarecrafters.io/xamarin/xamarin-forms-apps-nativas-introduccion

[31] S. Mostovoy, "Aplicación móvil multiplataforma, nativa y de código único para la publicación de contenidos gestionados desde web," España, 2015. [Online]. Available: https://repositorio.uam.es/handle/10486/668430

[32] "React js." [Online]. Available: https://es.reactjs.org

[33] J. G. G. Caballero, "¿cómo funciona react.js?" [Online]. Available: https://devcode.la/blog/como-funciona-reactj

[34] "Reactjs: Introducción y primeros pasos."

[35] "Visual studio code." [Online]. Available: https://code.visualstudio.com

[36] "Microsoft's cross-platform visual studio code app is based on technology from github's atom editor." [Online]. Available: https://thenextweb.com/apps/2015/04/30/microsofts-cross-platform-visual-studio-code-app-is-based-on-githubs-atom-editor/

[37] "Postgresql." [Online]. Available: https://www.postgresql.org/

[38] Anónimo, "Características, limitaciones y ventajas." [Online]. Available: http://postgresql-dbms.blogspot.com/p/limitaciones-puntos-de-recuperacion.html

[39] "Sqldeveloper." [Online]. Available: https://www.oracle.com/es/database/technologies/appdev/sql-developer.html

[40] "Keycloak." [Online]. Available: https://www.keycloak.org/index.html

[41] D. Naranjo, "Keycloak: una solución de gestión de acceso e identidad de código abierto," 2019. [Online]. Available: https://blog.desdelinux.net/keycloak-una-solucion-de-gestion-de-acceso-e-identidad-de-codigo-abierto

[42] "Jwt." [Online]. Available: https://jwt.io/

[43] Anónimo, "Autenticando una api con jwt." [Online]. Available: https://www.developerro.com/2019/03/12/jwt-api-authentication

[44] "Apisprout." [Online]. Available: https://github.com/danielgtaylor/apisprout

[45] "Netbeans." [Online]. Available: https://netbeans.org

[46] Calendamaia, "Netbeans." [Online]. Available: https://www.genbeta.com/desarrollo/netbeans-1

[47] "Maven." [Online]. Available: https://maven.apache.org

[48] J. Garzas, "Simple y rápido. entiende qué es maven." [Online]. Available: https://www.javiergarzas.com/2014/06/maven-en-10-min.html

[49] C. Alvarez, "¿qué es maven?" [Online]. Available: https://www.genbeta.com/desarrollo/que-es-maven

[50] J. L. P. Rojas, *Uaithne, Framework para el desarrollo de backend de aplicaciones.*, 2011th ed.

[51] J. F. León, "Evolución y aseguramiento de la calidad en uaithne e implantación en unit linked," España, 2014.

[52] "Tomcat." [Online]. Available: http://tomcat.apache.org/

[53] Anónimo, "Tomcat." [Online]. Available: https://es.wikipedia.org/wiki/Tomcat

[54] "jira software." [Online]. Available: https://www.atlassian.com/es/software/jira

Acrónimos

3NF Tercera Forma Normal.

API Application Programming Interface o Interfaz de Programación de Aplicaciones.

BBDD Base de datos.

BD Base de datos.

CSS Cascading Style Sheets o Hojas de estilo en cascada.

DNIe Documento Nacional de Identidad electrónico.

ER Entidad-Relacion.

HTML HyperText Markup Language o Lenguaje de Marcas de Hipertexto.

HTTPS Hypertext Transfer Protocol Secure o Protocolo Seguro de Transferencia de Hipertexto.

IDE Integrated Development Environment o Entorno de Desarrollo Integrado.

JSON JavaScript Object Notation.

JSP JavaServer Pages.

JWT JSON Web Token.

MPA Multi-Page Application o Aplicación Multi-Página.

MVC Model–View–Controller o Modelo-Vista-Controlador.

MVVM Model–View–ViewModel o Modelo-Vista-ModeloVista.

NFC Near-field communication o Comunicación de Campo Cercano.

PHP Hypertext Preprocessor o Preprocesador de Hipertexto.

POM Project Object Model.

POST Método de HTTP.

QR Quick Response code o Código de Respuesta Rápida.

REST Transferencia de Estado Representacional.

RF Requisito Funcional.

RNF Requisito No Funcional.

SAML Security Assertion Markup Language o Lenguaje de Marcado para Confirmaciones de Seguridad.

SPA Single-Page Application o Aplicación de Página Única.

SQL Structured Query Language o Lenguaje de Consulta Estructurada.

TFG Trabajo de Fin de Grado.

UAM Universidad Autónoma de Madrid.

UML Unified Modeling Language o Lenguaje Unificado de Modelado.

XHTML eXtensible HyperText Markup Language.

XSS Cross-Site Scripting.

YAML YAML Ain't Markup Language o YAML no es un lenguaje de marcado. Es un formato de serialización de datos.

Anexos

A

Diagramas

A.1 Diagrama de Casos de Uso

En este Apartado se muestra una imagen del Diagrama de Casos de Uso realizado durante el análisis de proyecto.

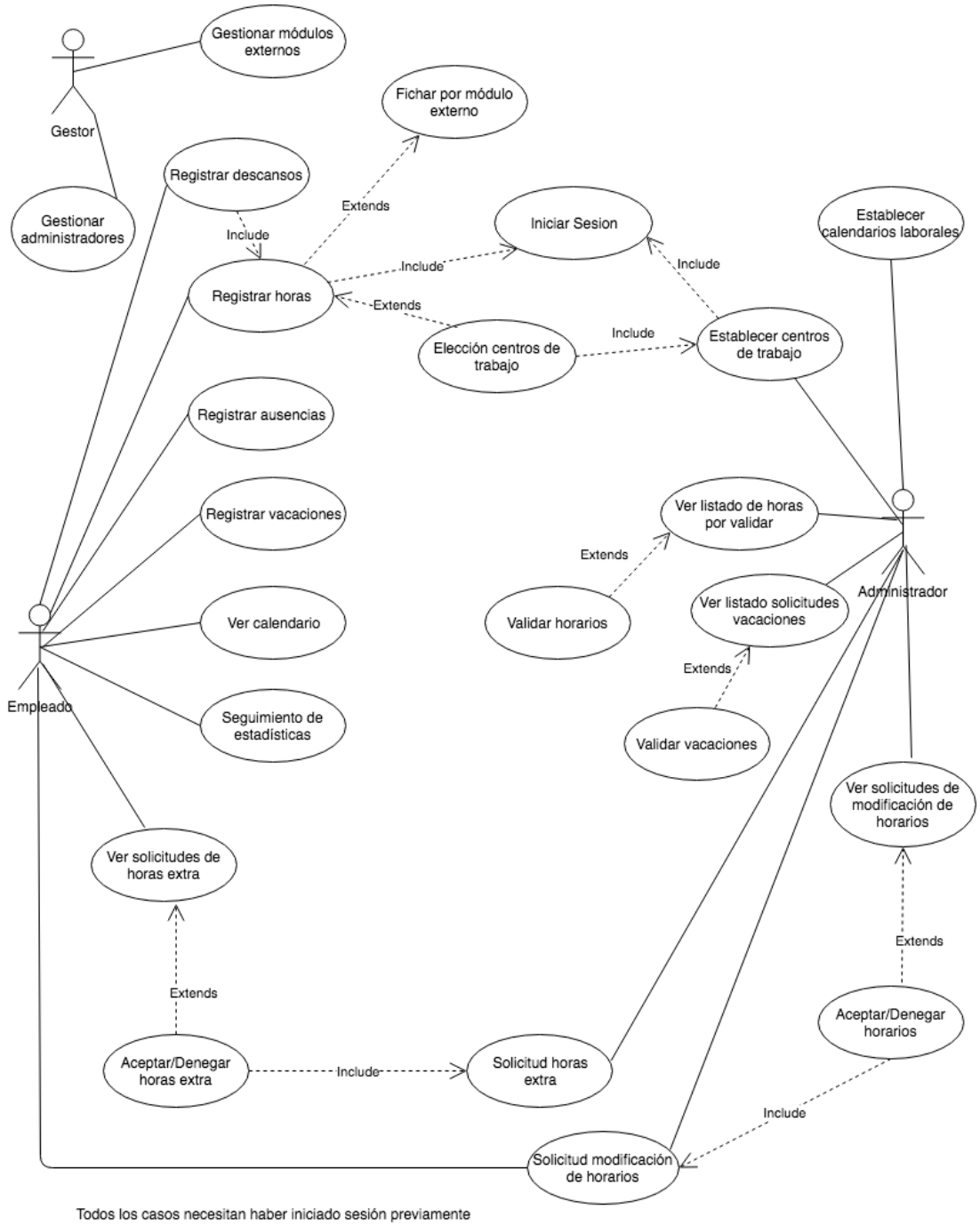

Figura A.1: Diagrama de Casos de Uso.

A.2 Diagrama Entidad-Relación

En este Apartado se muestra el Diagrama Entidad-Relación realizado durante el diseño de la base de datos.

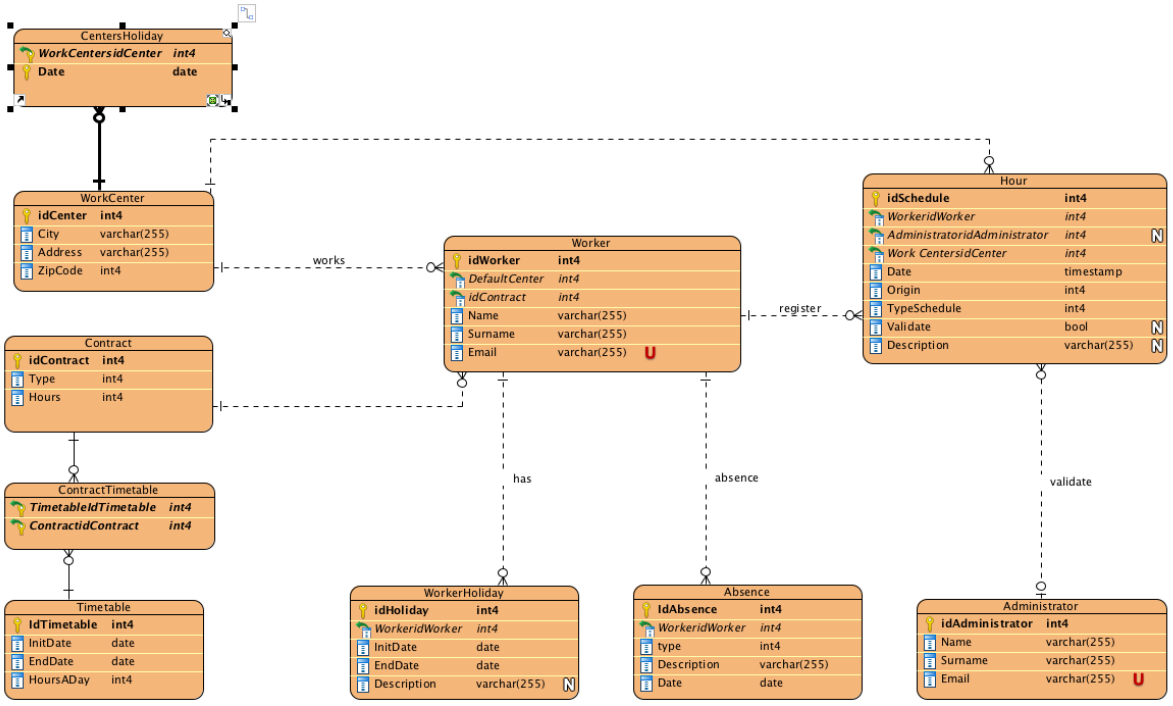

Figura A.2: Diagrama Entidad-Relación.

A.3 Diagrama de Secuencia

En este Apartado se muestra el Diagrama de Secuencia realizado para ver como sería una secuencia al añadir una hora en el sistema.

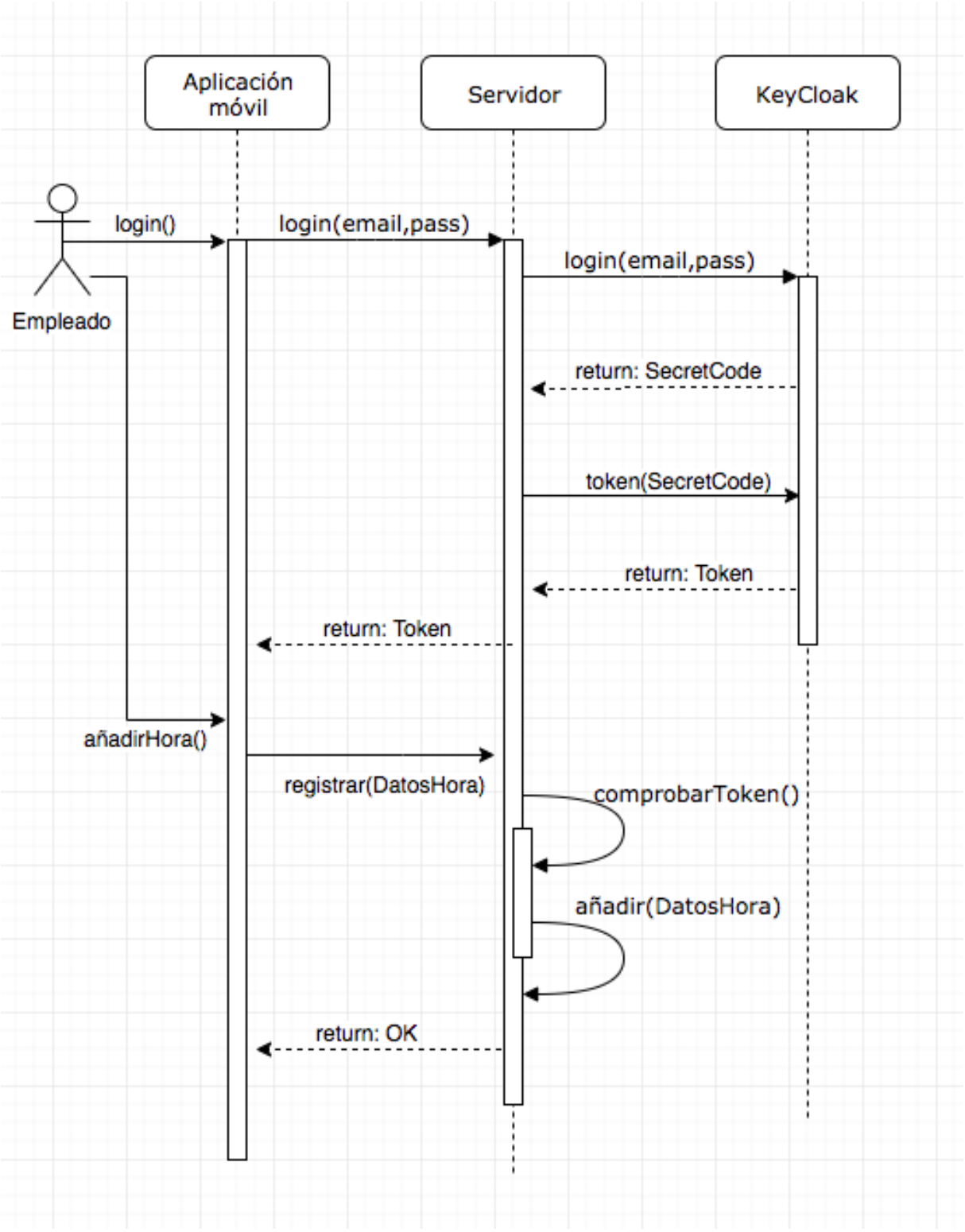

Figura A.3: Diagrama de secuencia.

B

Manual del programador

B.1 Front-end

B.1.1. React

En el siguiente ejemplo se muestra una declaración e inicialización de un componente en React. Además, también se ve el código para la redirección de las páginas, llamado *Router*

```
export default class BlogOverview extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      interval: null,
    };
  }
  render() {
    return (
      <Container fluid className="main-content-container␣px-4">
        {/* Page Header */}
        <Row>
          <Col lg="12" md="12" sm="12" className="mb-4">
            <CustomTimeline />
          </Col>
        </Row>
      </Container>
    );
  }

}
```

Código B.1: Componente de React

```
<Router basename={process.env.REACT_APP_BASENAME || ""}>
    <div>
      {routes.map((route, index) => {
        return (
          <Route
            key={index}
            path={route.path}
            exact={route.exact}
            component={withTracker(props => {
              return (
                <route.layout {...props}>
                  <route.component {...props} />
                </route.layout>
              );
            })}
          />
        );
      })}
    </div>
  </Router>
```

Código B.2: Router de React

B.1.2. Xamarin

En este Apartado se muestra un ejemplo de como se crean distintos componentes en Xamarin. En este caso son un *BoxView*, *Grid*, y un *DatePicker*.

```
Grid grid = new Grid
            {
                RowSpacing = 0,
                ColumnSpacing = 0,
                RowDefinitions = {
                    new RowDefinition { Height =50 },
                    new RowDefinition { Height =50 },
                    new RowDefinition { Height = new GridLength ↩
                       (1, GridUnitType.Star) },
                },
                ColumnDefinitions = {
                    new ColumnDefinition { Width =  new ↩
                       GridLength (1, GridUnitType.Auto) },
                    new ColumnDefinition { Width =  new ↩
                       GridLength (1, GridUnitType.Star) },
                },
            };
            BoxView box = new BoxView
            {
                BackgroundColor = Color.FromHex("0D9BE5"),
                HorizontalOptions = LayoutOptions.FillAndExpand,
```

```
                VerticalOptions = LayoutOptions.FillAndExpand,
                Margin = 0,

            };
            grid.Children.Add(box, 0, 0);
            Grid.SetRowSpan(box, 2);

             _datePicker = new DatePicker
            {
                Date = DateTime.Today,
                Format = "ddd dd MMM",
                HorizontalOptions = LayoutOptions.Center,
                VerticalOptions = LayoutOptions.CenterAndExpand,
                FontSize = 40,

                TextColor = Color.White,
                FontAttributes = FontAttributes.Bold,
                BackgroundColor = Color.Transparent,
            };



            _datePicker.DateSelected +=(sender, e) =>
            {

                UpdateHours();
            };

            grid.Children.Add(_datePicker, 0, 0);
            Grid.SetRowSpan(_datePicker, 2);
            Grid.SetColumnSpan(_datePicker, 2);
```

Código B.3: Composición de componentes en Xamarin

En el código siguiente se puede ver el método para realizar peticiones en el servidor. Se serializa el objeto que se manda en JSON y lo que devuelve se deserializa en el objeto que se le ha indicado.

```
try
            {
                HttpContent sendContent;

                if (byteContentType)
                {
                    Byte[] sendBody = Encoding.UTF8.GetBytes(↩
                       JsonConvert.SerializeObject(arguments, ↩
                       Formatting.None, new ↩
                       JsonSerializerSettings { NullValueHandling↩
                        = NullValueHandling.Ignore/*, ↩
                       ContractResolver = new ↩
                       CamelCasePropertyNamesContractResolver()*/↩
```

```
                }))).ToArray<Byte>();

            sendContent = new ByteArrayContent(sendBody);
            sendContent.Headers.Add("Content-Type", "↩
                application/json; charset=UTF-8");
        }
        else
        {
            sendContent = new MultipartFormDataContent();

            var multipartContent = sendContent as ↩
                MultipartFormDataContent;
            foreach (KeyValuePair<string, object> pair in↩
                arguments)
            {
                multipartContent.Add(new StringContent(↩
                    pair.Value.ToString(), Encoding.UTF8),↩
                    pair.Key);
            }
        }

        var oauth = await SecureStorage.GetAsync("↩
            oauth_token");
        if (oauth != null)
        {
            _client.DefaultRequestHeaders.Authorization =↩
                new AuthenticationHeaderValue("Bearer", ↩
                oauth);
        }

        HttpResponseMessage response = await _client.↩
            PostAsync(uri, sendContent);

        if (response.IsSuccessStatusCode)
        {
            var content = await response.Content.↩
                ReadAsStringAsync();
            var token = JsonConvert.DeserializeObject<T>(↩
                content);
            return token;
        }
        else if (response.StatusCode == HttpStatusCode.↩
            Unauthorized)
        {


            await Xamarin.Forms.Application.Current.↩
                MainPage.DisplayAlert("Su sesi n ha↩
                caducado", "Por favor, vuelva a↩
```

```
                    autentificar su usuario", "Ok");

                Xamarin.Forms.Application.Current.MainPage = ↩
                    new LoginPage();
            }
            return default(T);

        }
        catch (Exception ex)
        {
            return default(T);
        }
```

Código B.4: Método para la petición al servidor

B.2 Back-end

B.2.1. Uaithne

En este Apartado se enseña una clase con una serie de objetos de *Uaithne*. Se declaran operaciones directas en base de datos, que te genera automáticamente. *Entity* es el componente que se utiliza como clase para acceder a la base de datos.

```
@OperationModule
@MyBatisMapper
public class _hour {
    @MappedName("hour")
    @Entity
    class _HourDataBase{
        @Id
        @MappedName("idschedule")
        int idSchedule;
        @MappedName("comments")
        String comments;
        @MappedName("origin")
        int origin;
        @MappedName("reasontype")
        int reasonType;
        @MappedName("idcenter")
        int idCenter;
        @MappedName("datetime")
        Date dateTime;
        @MappedName("worker_id")
        int idPerson;
    }

    @SelectMany(result = _HourDataBase.class)
    @CustomSqlQuery(where = {
```

```
        "CAST(datetime␣as␣date)␣=␣#{dateTime,␣jdbcType=DATE}"
    }, orderBy = {"datetime␣asc"})
    class _SelectUserHours{
        @MappedName("worker_id")
        int idPerson;
        @MappedName("datetime")
        Date dateTime;
    }

}
```

Código B.5: Módulo de operación de Uaithne

En el siguiente código, se muestra el bus de operaciones en Uaithne. Existen dos capas, la de base de datos y la de operaciones. El bus pasa por cada uno de los ejecutores buscando la operación a ejecutar.

```
        MappedExecutorGroup dbLayer = new MappedExecutorGroup();
        dbLayer.addExecutor(new CenterMapper(sessionProvider));
        dbLayer.addExecutor(new HourMapper(sessionProvider));

        ChainedMappedExecutorGroup backendLayer = new ↩
           ChainedMappedExecutorGroup(dbLayer);
         backendLayer.addCustomExecutor(new UserExecutorImpl(↩
            sessionProvider));

        ManagedSqlSessionExecutorGroup ↩
           managedSqlSessionExecutorGroup = new ↩
           ManagedSqlSessionExecutorGroup(backendLayer, ↩
           sessionProvider);
```

Código B.6: Bus de Operaciones de Uaithne

B.2.2. Servidor

En este Apartado, se muestra un ejemplo de una notación para para las peticiones en el servidor siguiendo el estándar de OpenApi.

```
    @POST
    @Path("/Add")
    @Consumes({ "application/json" })
    @Produces({ "application/json" })
    @io.swagger.annotations.ApiOperation(value = "", notes = "", ↩
       response = GenericResponse.class, tags={  })
    @io.swagger.annotations.ApiResponses(value = {
        @io.swagger.annotations.ApiResponse(code = 200, message =↩
            "Respuesta␣para␣cuando␣se␣registra␣bien", response = ↩
           GenericResponse.class) })
```

```
    public Response hourAddPost(@ApiParam(value = "" ,required=↩
       true) @NotNull @Valid AddHourRequest addHourRequest,↩
       @Context SecurityContext securityContext);
```

Código B.7: Ejemplo de petición en el servidor

B.2.3. KeyCloak

En este Apartado se enseña el acceso para obtener el token de Keycloak a partir de un email y su contraseña.

```
RSATokenVerifier verifier = RSATokenVerifier.create(token);
PublicKey publicKey = getRealmPublicKey(keycloak, verifier.↩
   getHeader());
return verifier.realmUrl(getRealmUrl()) //
          .publicKey(publicKey) //
          .verify() //
          .getToken();
      } catch (VerificationException e) {
return null;
      }
    }
private KeycloakBuilder newKeycloakBuilderWithPasswordCredentials↩
   (String username, String password) {
return newKeycloakBuilderWithClientCredentials() //
        .username(username) //
        .password(password) //
        .grantType(OAuth2Constants.PASSWORD);
    }
private KeycloakBuilder newKeycloakBuilderWithClientCredentials()↩
    {
return KeycloakBuilder.builder() //
        .realm(realmId) //
        .serverUrl(serverUrl)//
        .clientId(clientId) //
        .clientSecret(clientSecret) //
        .grantType(OAuth2Constants.CLIENT_CREDENTIALS);
    }
```

Código B.8: Ejemplo de petición en el servidor

C

Diseño Final

En este Anexo se muestran algunos ejemplos de como ha quedado el diseño final de las aplicaciones.

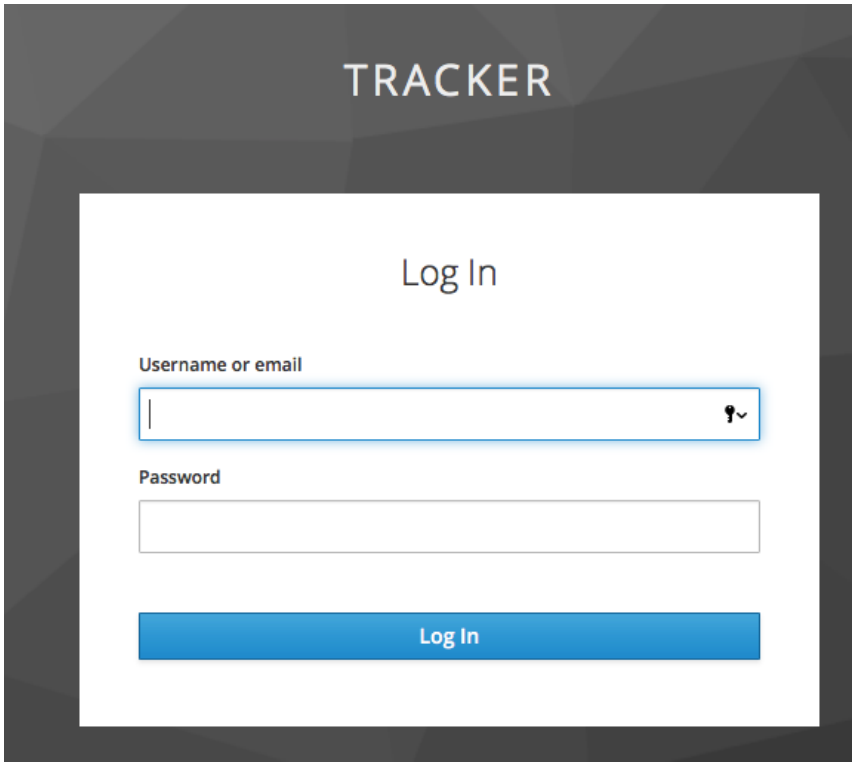

Figura C.1: Login de las aplicaciones.

C.1 Aplicación móvil

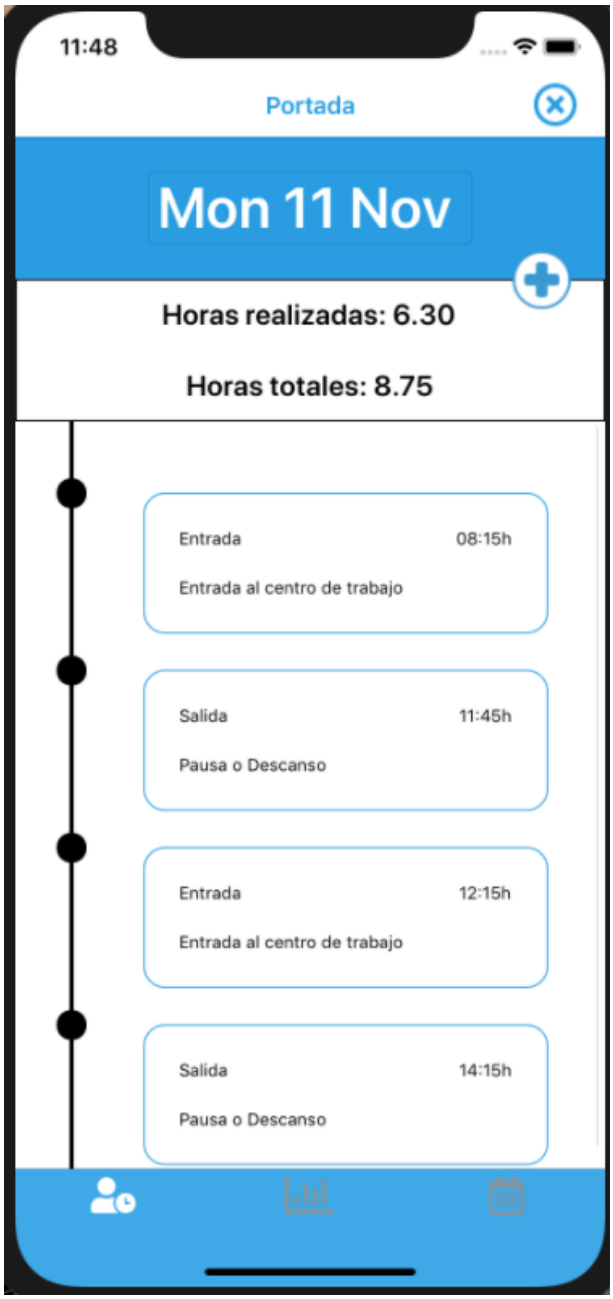

Figura C.2: TimeLine de las horas en la aplicación.

Figura C.3: Pantalla para añadir las horas.

C.2 Aplicación Web

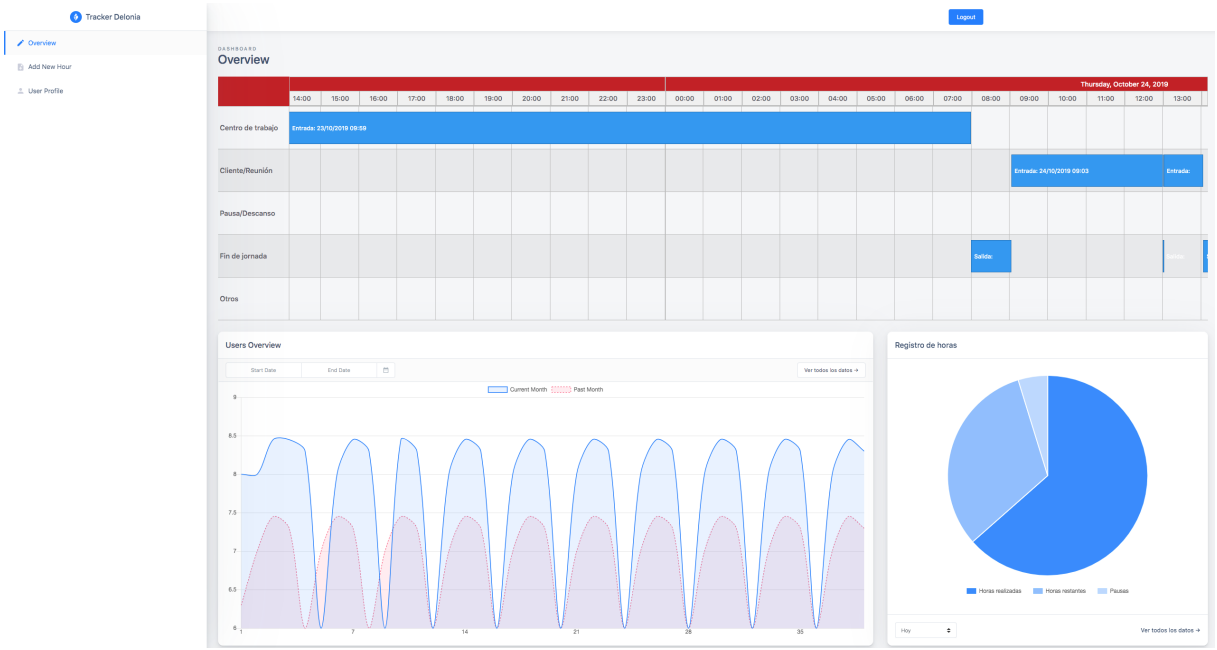

Figura C.4: Pantalla principal de la web.

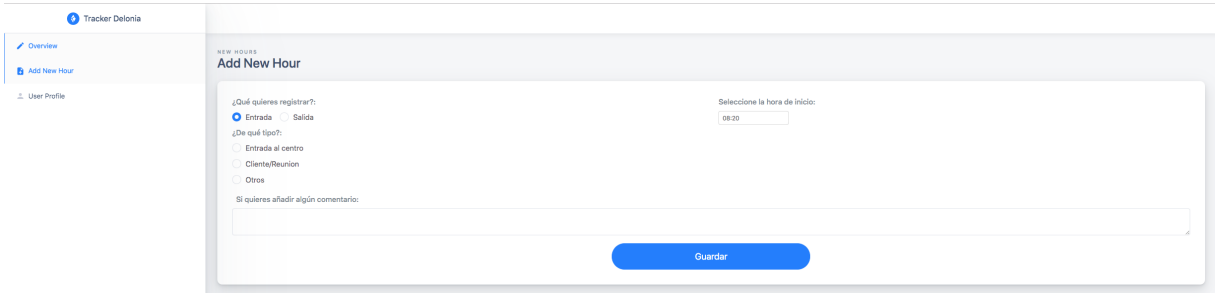

Figura C.5: Pantalla para añadir las horas desde la web.